НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-додаток для підтримки формування раціону користувача на основі платформи AWS»

Виконала:
студентка IV курсу, групи КП-61
Карпенко Олена Олександрівна _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська Марина Віталіївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Карпенко Олені Олександрівні

1. Тема проєкту «Веб-додаток для підтримки формування раціону користувача на основі платформи AWS», керівник проєкту Заболотня Тетяна, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. № 1181-с

2. Термін подання студентом проєкту «15» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - обґрунтування вибору засобів реалізації;
   - структурно-алгоритмічна організація;
   - особливості реалізації програмного забезпечення;

5. Перелік обов'язкового графічного матеріалу:
   - структура бази даних (креслення);
   - алгоритм оброблення рецептів (креслення);
   - архітектура веб-додатку (плакат);
   - діаграма прецендентів (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2019	
2.	Розроблення та узгодження технічного завдання	28.11.2019	
3.	Розроблення структури веб-додатку	15.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2020	
7.	Програмна реалізація веб-додатку	10.03.2020	
8.	Тестування веб-додатку	17.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	11.04.2020	
11.	Підготовка графічної частини дипломного проєкту	21.04.2020	
12.	Оформлення документації дипломного проєкту	26.05.2020	

Студент Олена КАРПЕНКО

Керівник проєкту Тетяна ЗАБОЛОТНЯ

АНОТАЦІЯ

Даний дипломний проект присвячений створенню веб-додатку для підтримки формування раціону користувача на основі платформи AWS.

Веб-додаток являє собою сайт призначений для надання можливості відвідувачам продивлятися список збережених в його базі даних рецептів та надання користувачам рекомендацій рецептів, керуючись наданими ними відомостями про особливості раціону та потреби у поживних речовинах. В даній роботі проведено аналіз сучасних існуючих програмних рішень для формування раціону користувачів. На основі отриманих даних було зроблено висновок про актуальність даного дослідження. Розглянуто сучасні технології для побудови веб-застосунків та обрано засоби для реалізації даного веб-додатку. Обґрунтовано доцільність розроблення веб-додатку на основі платформи хмарних обчислень AWS. Описано особливості проектування сервіс-орієнтованої архітектури веб-додатку, її переваги та недоліки у поєднанні із без серверною моделлю обчислень.

У даному дипломному проєкті розроблено: архітектуру веб-додатку, алгоритми авторизації користувачів, збору та обробки даних рецептів із тематичних веб-сайтів, генерації пропозицій рецептів користувачам а також графічні елементи та дизайн сторінок.

Веб-додаток для підтримки формування раціону користувача на основі платформи AWS дозволяє користувачам підтримувати оптимальний баланс поживних та корисних речовин у організмі за допомогою рекомендацій рецептів та пошуку серед них за критеріями.

ABSTRACT

This diploma project deals with the development of a web application to support the development of a user's diet based on the AWS platform.

The web application is a site designed to enable visitors to view a list of recipes stored in its database and to provide users with recipe recommendations based on the information they have provided about diet features and nutrient requirements. In this work, an analysis of current existing software solutions for the formation of users' diets is conducted. On the basis of the data obtained, it was concluded that this study is relevant. Modern technologies for building web applications have been considered and the means for implementing this web application have been selected. The feasibility of developing a web application based on the cloud computing platform AWS. Features of service-oriented web-application architecture design, its advantages and disadvantages in combination with the serverless model of computing are described.

This diploma project has been developed: web application architecture, user authorization algorithms, recipe data collection and processing from thematic websites, generation of recipe offers to users, as well as graphic elements and page design.

The web application to support the formation of a user's diet based on the AWS platform allows users to maintain an optimal balance of nutrients and nutrients in the body through recipe recommendations and search among them by criteria.

АННОТАЦИЯ

Данный дипломный проект посвящен созданию веб-приложения для поддержки формирования рациона пользователя на основе платформы AWS.

Веб-приложение представляет собой сайт, предназначенный для предоставления возможности посетителям просматривать список сохраненных в его базе данных рецептов и предоставления пользователям рекомендаций рецептов, руководствуясь предоставленными ими сведениям об особенностях рациона и потребности в питательных веществах. В данной работе проведен анализ современных существующих программных решений для формирования рациона пользователей. На основе полученных данных был сделан вывод об актуальности данного исследования. Рассмотрены современные технологии для построения веб-приложений и выбранные средства для реализации данного веб-приложения. Обоснована целесообразность разработки веб-приложения на основе платформы облачных вычислений AWS. Описаны особенности проектирования сервис-ориентированной архитектуры веб-приложения, ее преимущества и недостатки в сочетании с без серверной моделью вычислений.

В данном дипломном проекте разработано: архитектура веб-приложения, алгоритмы авторизации пользователей, сбора и обработки данных рецептов из тематических сайтов, генерации предложений рецептов пользователям, а также графические элементы и дизайн страниц.

Веб-приложение для поддержки формирования рациона пользователя на основе платформы AWS позволяет пользователям поддерживать оптимальный баланс питательных и полезных веществ в организме с помощью рекомендаций рецептов и поиска среди них по критериям.

ДП.045440-01-90 Веб-додаток для підтримки формування раціону користувача на основі платформи AWS. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Веб-додаток для	5	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Технічне завдання		
ДП.045440-03-81	Веб-додаток для	70	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Пояснювальна		
	записка		
ДП.045440-04-51	Веб-додаток для	4	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Програма та		
	методика тестування		
ДП.045440-05-34	Веб-додаток для	10	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Веб-додаток для	1	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Структура бази		
	даних. ERD діаграма		
ДП.045440-07-99	Веб-додаток для	1	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Функціональні		
	можливості користувача.		
	Діаграма прецедентів		
ДП.045440-08-98	Веб-додаток для	1	
	підтримки формування		
	раціону користувача		
	на основі платформи		
	AWS. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"
Науковий керівник кафедри
_____ Іван ДИЧКА
"___" _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олена КАРПЕНКО

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-додаток для підтримки формування раціону користувача на основі платформи AWS.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання користувачами, які потребують зручного автоматичного засобу для контролю свого раціону та підбору рецептів, які відповідають їх потребам у речовинах та калоріях.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-додаток повинен забезпечувати такі основні функції:

1) наповнення бази даних рецептів за допомогою збору даних із тематичних веб-сайтів;
2) можливість авторизації користувачів;
3) можливість пошуку рецепту за критеріями: країна походження, інгредієнти, тип страви, вміст БЖВ та калорійність;
4) можливість авторизованим користувачем отримувати рекомендації рецептів страв на основі відомостей про нього.

Розробку виконати на платформі Microsoft .Net та AWS з використанням технології React.js.

Додаткові вимоги:

1) реалізація безперервної доставки та інтеграції ПЗ;
2) опис інфраструктури веб-додатку за допомогою підходу IaC;
3) перегляд рецепту користувачами;
4) можливість користувачем залишити відгук про рецепт;
5) пошук серед рецептів за назвою.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:

– «Функціональні можливості користувача. Діаграма прецедентів»;
– «Структура бази даних. ERD-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2019

Розроблення та узгодження технічного завдання……....28.11.2019

Розроблення структури веб-додатку…….....15.12.2019

Розроблення дизайну сторінок та графічних елементів……..03.02.2020

Програмна реалізація веб-додатку…….....17.03.2020

Тестування веб-додатку…….....03.04.2020

Підготовка матеріалів текстової частини проєкту……..….28.04.2020

Підготовка матеріалів графічної частини проєкту……...…12.05.2020

Оформлення технічної документації проєкту……..…25.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олена КАРПЕНКО

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

ОС – операційна система.

ООП – об’єктно-орієнтоване програмування.

Веб-сайт – сукупність веб-сторінок, які доступні у мережі Інтернет, які об’єднані за змістом та за навігацією під одним доменним ім’ям.

Веб-браузер – програмне забезпечення для переглядів веб-сайтів у мережі Інтернет.

DOM – об’єктна модель документа, представлення HTML-документів у вигляді дерева тегів.

HTML – стандартна мова розмітки гіпертекстових документів в мережі Інтернет.

CSS – каскадні таблиці стилів, формальна мова яка використовується для опису оформлення зовнішнього вигляду документу, створеного з використанням мови розмітки HTML.

Фреймворк – програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, яке полегшує поставлене перед розробником завдання і об'єднує різні компоненти великого програмного проєкту.

SPA – Single Page Application (укр. «односторінковий застосунок»), веб-додаток, який використовує єдиний HTML-документ, як оболонку для всіх веб-сторінок, та який забезпечує взаємодію з користувачем через динамічне підкачуємі HTML, CSS, JavaScript.

JSX – надбудова на JavaScript, яка дозволяє використовувати XML-подібний синтаксис в JavaScript.

Webpack – інструмент, який дозволяє компілювати JavaScript модулі в єдиний JS-файл.

БД – база даних.

СУБД – система управління базами даних, програмне забезпечення для керування даними.

ACID – спеціальні вимоги до СУБД, що гарантують надійну роботу транзакцій: Atomicity (укр. «атомарність»), Consistency (укр. «узгодженість»), Isolation (укр. «ізольованість»), Durability (укр. «довговічність»).

Транзакція – набір операцій в базі даних, що є логічною одиницею роботи з даними.

JSON – текстовий формат обміну даними, що базується на мові програмування JavaScript.

HTTP – Hypertext Transfer Protocol (укр. «протокол передачі гіпер-текстових документів»), протокол прикладного рівня для передачі даних у комп'ютерних мережах.

HTTPS – Hypertext Transfer Protocol Secure (укр. «протокол передачі гіпер-текстових документів»), розширення протоколу HTTP для підтримки шифрування з метою повищення безпеки.

API – Application Programming Interface (укр. «інтерфейс прикладного програмування») – набір визначень, протоколів та інструментів для розробки ПО та додатків.

REST – Representational State Transfer (укр. «передача репрезентативного стану»), архітектурний стиль взаємодії компонентів розподіленого додатку у мережі.

ВСТУП

Важливу роль в житті будь-якої людини відіграє здорове та збалансоване харчування. Однак, сучасний швидкий ритм життя обмежує час, який ми можемо виділити на планування свого раціону. Як наслідок, багато людей страждають від зниження рівня активності та розумової діяльності, передчасного старіння та скорочення тривалості життя.

Можливим варіантом виходу із ситуації, що склалася, є користування послугами дієтологів. Але такі консультації майже завжди мають високу вартість і не є доступними для всіх верств населення. Разом з тим, багатьом людям для покращення збалансованості свого харчування достатньо було б отримати базові рекомендації щодо необхідного набору мікроелементів та продуктів або страв, що їх містять. Такого роду інформаційну підтримку може надавати і відповідне ПЗ. Тому автоматизація підбору рекомендацій щодо дієти для людини на основі її вподобань та потреб є актуальною задачею.

З огляду на вищезазначене, об’єктом даного дослідження став процес автоматизованого формування порад щодо корегування індивідуального щоденного раціону людини, а предметом дослідження – програмні засоби для підтримки пошуку рецептів страв та отримання рекомендацій щодо збалансування харчування користувачів. Основними перевагами ресурсу є зручна система рекомендацій рецепту за вподобаннями та потребами користувача. У великій базі, яка регулярно поповнюється, однозначно знайдеться рецепт, який піддійте до смаку навіть найпримхливішому користувачеві.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яка вирішується ПЗ

Основною проблемою сучасного світу є брак часу. Через високий темп життя у 21 столітті, пересічній людині дедалі важче стає виділити час на приготування корисної та поживної їжі. Питання харчування часто вирішують замовленням страв із ресторанів. В останні роки широку популярність набули сервіси доставки їжі, такі як Glovo та Uber Eats. Однак це не означає, що люди замовляють корисні страви. Навпаки, зазвичай вони купують фаст-фуд. Це негативно вливає на їх здоров'я та фізичний стан, викликаючи проблеми із зайвою вагою.

Проблемою, яка змушує людей шкодити своєму організму – відсутність зручного методу пошуку рецептів, які б задовольняли всім критеріям, за якими користувач зазвичай їх шукає:

- складність приготування рецепту та навички головуванні, які знадобляться;
- час, необхідний для приготування;
- рейтинг рецепту і коментарі від відвідувачів ресурсу із рецептом;
- якісні та зрозумілі інструкції;
- ціна інгредієнтів;
- розрахунок калорійності готової страви.

Готувати корисно і смачно - бажання багатьох людей, які змушені від цього відмовитися через такі фактори:

1. Брак часу на вивчення і пошук нових рецептів. Люди хочуть куштувати щось нове. Проте в реаліях сучасного світу часу на експерименти майже не лишається. Тому більшість спроб віднайти щось нове призводять до повернення до використання добре відомих рецептів, які є перевіреними часом.
2. Брак необхідних інгредієнтів до рецепту. Ситуація, коли цікавий рецепт знайдено і заплановано його приготування, проте в

останній момент виявляється, що чогось бракує – не є рідкою. Зазвичай, люди просто відмовляються від своїх намірів і готують щось інше, необов'язково за новим рецептом.

3. Відсутність інформації про поживну складову страви. Для спортсменів є важливим підрахунок білків, жирів та вуглеводів. Однак більшість веб-ресурсів не приділяють цій інформації потрібної уваги, що змушує відвідувачів відмовитися від їх використання.

Ідея слідкувати за своїм раціоном не є новою. Саме тому існує безліч аналогів та ресурсів, які допомагають людям слідкувати за їх дієтою. Зазвичай вони мають наступну модель монетизації – платна підписка на особистого дієтолога із рекомендаціями щодо страв. Однак ці рецепти є запрограмованими та не змінюються із часом для кожного клієнта, враховуючи його вподобання.

Враховуючи вищенаведене, можна сказати, що проблема, яку має вирішувати розроблюваний сервіс – допомога людям у пошуку рецептів страв, які будуть задовільнять їх потреби.

В рамках виконання даного дипломного проєкту було проаналізовано вже існуючі рішення та знайдено їх переваги і недоліки. На основі зібраних даних було сформовано вимоги до програмного забезпечення, розроблюваного в рамках дипломного проєкту.

1.2. Аналіз існуючих рішень

1.2.1. *Лікар - дієтолог*

Дієтолог – фахівець в галузі дієтології, науки про біохімічні та фізіологічні основи харчування людини [1]. Його основна ціль – розробити щоденний план харчування, який включає в себе комплекс вітамінів та мінералів, на основі індивідуальних фізіологічних особливостей людини та її загального стану здоров'я. Дієтолог може лікувати пацієнтів із проблемами зайвої або недостатньої ваги, допомагати лікарям всіх

спеціальностей складати раціон для швидшого одужання їх пацієнтів. Очевидно, що для винесення найбільш точного діагнозу, цей лікар вимагає від пацієнтів здачі аналізів та особистої зустрічі для оцінки фізичних показників перед своєю роботою над раціоном клієнта.

Перевагами звернення до дієтолога є:

- професійна консультація;
- винесення точного діагнозу;
- формування раціону, який має найбільшу відповідність для кожного пацієнта.

Очевидними недоліками є:

- платна консультація в медичних установах;
- потреба в особистій зустрічі;
- сформований раціон, як правило, не містить опису рецептів, за яким слід готувати страви, а містить лише перелік інгредієнтів, які слід вживати пацієнту, як результат - все ще існує потреба у пошуку рецептів.

Крім професійних лікарів-дієтологів, зараз набирають популярності форуми та групи, в яких адміністратори нав’язують іншим людям свої програми схуднення та здорового образу життя. Таких людей прийнято називати «дієтологами» – любителями. Це люди, які можуть не мати відповідної освіти в області дієтології, проте вони все одно надають послуги щодо розроблення раціону, який може зовсім не підходити їх клієнтам. Для цього вони використовують соціальні мережі, такі як Instagram, Facebook та Telegram. Зазвичай, дієтолог-любитель володіє основами дієтології, проте не може бути заміною справжнього лікаря, а тим паче брати за свою консультацію гроші та наполягати на своїх порадах, або нав’язувати покупку біодобавок в якості ліків.

Обізнані люди намагаються уникати таких людей та не виявляють довіри ресурсам, які намагаються продати програми раціону своїм користувачам, не маючи на це справжніх сертифікатів. Тому сервіс,

розроблений в рамках даного дипломного проєкту, не буде намагатися виносити діагноз своїм користувачам або нав’язувати їм рецепти в якості програми схуднення тощо. Даний сервіс має стати наступним кроком, до якого звертаються користувачі після того, як вони отримали рекомендації справжнього дієтолога або знають свої потреби. Він буде допомагати їм знайти рецепт страви на основі даних про інгредієнти та мікроелементи.

1.2.2. *Cara: Food, Mood, Poop Tracker*

Cara: Food, Mood, Poop Tracker – мобільний додаток, розроблений за підтримки лікарів [2]. Він автоматично вираховує потрібний для користувача план харчування, ґрунтуючись на його цілях та початкових біометричних даних таких як: вага, зріст, стать та вік. Широкий функціонал додатку вигідно виділяє його серед конкурентів. Основною перевагою є велика база рецептів, яка регулярно поповнюється вручну, онлайн-лічильник калорій, широка база продуктів з описом їх характеристик і калорійності.

До переваг цього додатку відносяться:

- поради реальних лікарів;
- велика база даних про продукти;
- мобільний інтерфейс;
- візуалізація статистичних даних про раціон користувача, їх аналітика;
- пошук залежності між самопочуттям та раціоном користувача.

Однак даний сервіс не надає можливості гнучкого пошуку рецептів та обмежується пошуком рецептів за назвою та переліком інгредієнтів. В ньому відсутні пошук за складністю, кількості рецензій та назвою кухні світу.

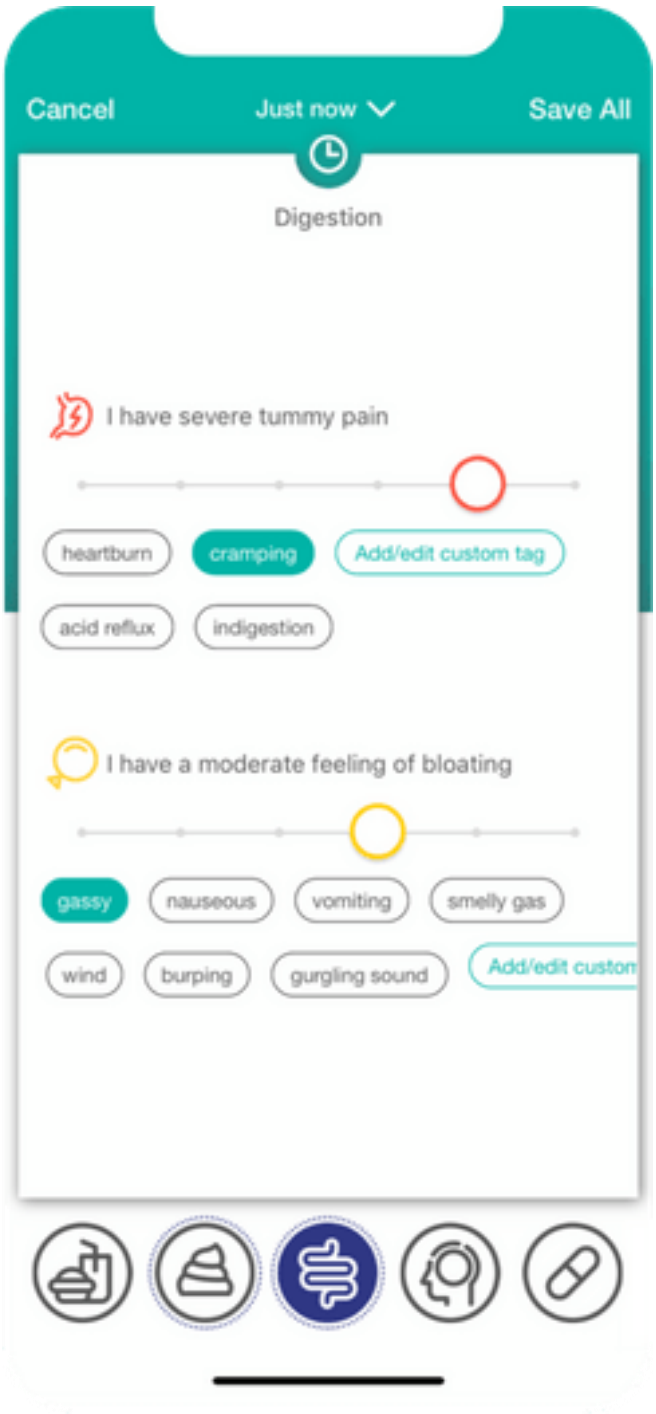


Рис. 1.1. Інтерфейс мобільного додатку Cara: Food, Mood, Poop Tracker [2]

1.2.3. *Rise Up + Recover*

Rise Up + Recover – багатофункціональний щоденник харчування [3]. Додаток надає можливість записувати всі прийоми їжі, програми тренувань та кількість спожитої води. Основною перевагою даного рішення є нагадування про прийом їжі та води, ведення календаря із даними про їх споживання. В кінці місяця користувач може бачити графік із даними про

його харчування, однак висновки із отриманої інформації він повинен робити сам.

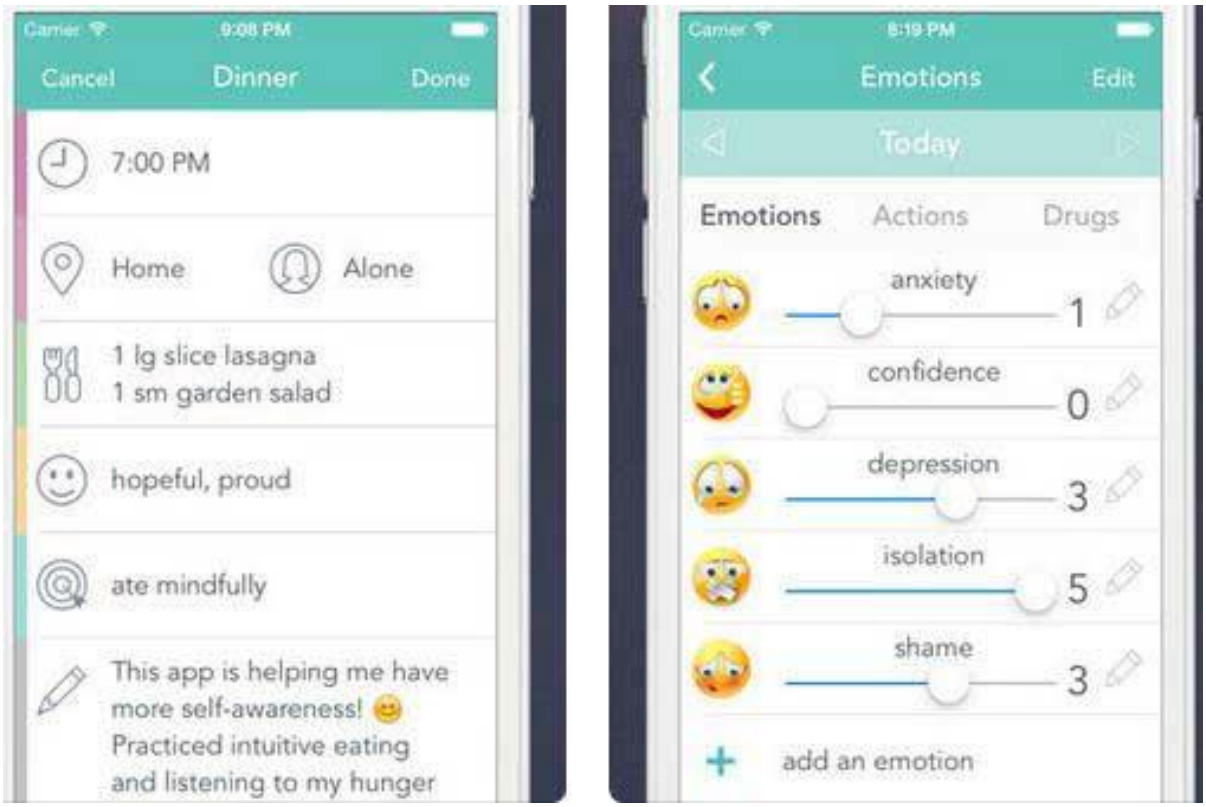


Рис. 1.2. Інтерфейс мобільного додатку Rise Up + Recover [3]

Очевидним недоліком цього додатку є обмежений функціонал, який полягає лише у веденні щоденника без можливості шукати або отримувати рецепти разом із нагадуванням про прийом їжі.

1.2.4. *Веб-сайт «Еда»*

Веб-сайт «Еда» – ресурс у мережі Інтернет, який представляє собою базу рецептів, інгредієнтів та кулінарних інструментів [4]. Користувачі мають можливість шукати рецепти за багатьма критеріями, серед яких: час приготування, продукти-складові, країна походження тощо. Кожен рецепт має чітко сформовану покрокову інструкцію приготування із фото або відео проміжного стану страви та опис поживних складових кінцевої страви. Окрім цього, зареєстровані користувачі можуть голосувати за рецепт та

залишати свої коментарі. Формат подачі інформації на веб-сайті схожий на журнал: на головній сторінці розміщено статі від редакторів, які несуть інформацію про добірку рецептів на кожен випадок життя.

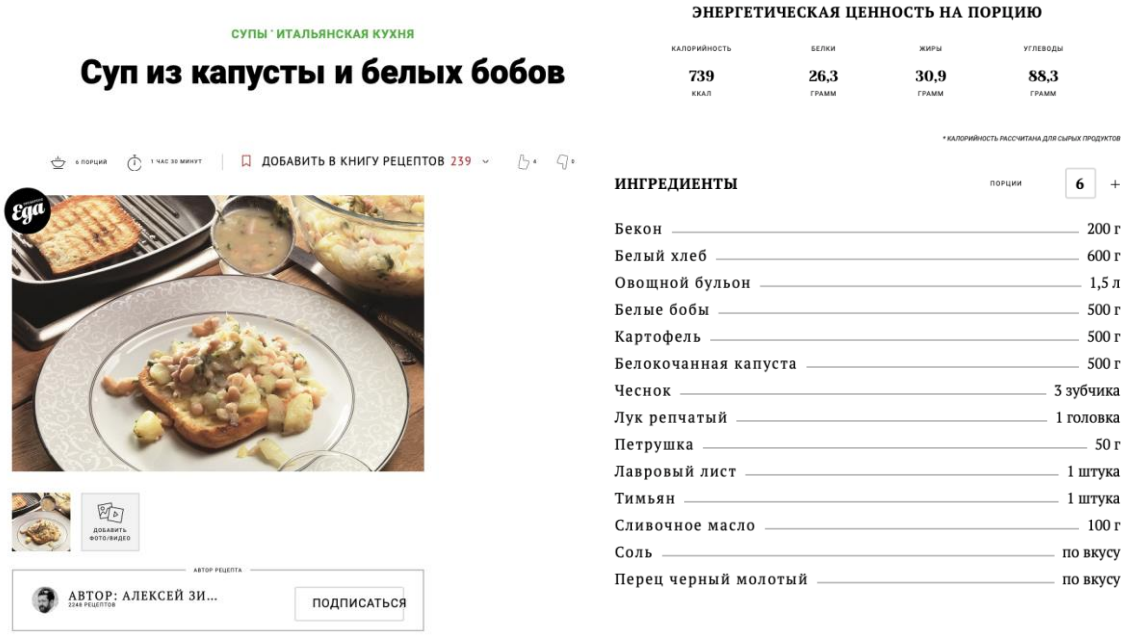


Рис. 1.3. Веб-сторінка із рецептом страви на веб-сайті «Еда» [4]

Перевагами цього сервісу є:

- найбільша база рецептів серед перелічених ресурсів;
- зручний інтерфейс;
- детальний опис потрібних інгредієнтів;
- інформація про калорійність, мікроелементи та час на приготування страви;
- детальні покрокові інструкції із фото результату кожного кроку;
- розрахунок маси інгредієнтів із урахуванням бажаної кількості порцій;
- добірки статей від редакторів;
- можливість вести книгу рецептів.

До недоліків можна віднести:

- персоналізовані рекомендації користувачам, врахування його фізіологічних особливостей.
- відсутність ведення статистики приготованих страв користувачем.
- велику кількість реклами.

1.3. Результати проведеного аналізу

На основі результатів проведеного аналізу встановлено, що описані рішення повністю не вирішують задачу допомоги та мотивації користувачів готувати смачно та корисно, шукати нові рецепти. Тому розроблюваний додаток повинен реалізувати всі найкращі практики, які описані вище.

Сформовано список критеріїв, за якими можна оцінити кожен із приведених додатків та практик:

- наявність бази рецептів, які готові до приготування;
- наявність персоналізованих рекомендації;
- наявність візуалізації статистики користувача;
- наявність можливості залишати відгуки до рецептів;
- наявність пошуку рецептів за критеріями.

На основі цих критеріїв оцінено кожне із рішень (табл. 1.1). Виходячи з отриманих результатів, сформовано список вимог до сервісу, який розроблено в рамках даного дипломного проєкту:

- має бути зібрана база рецептів;
- рецепти мають містити дані про калорійність, поживні речовини, час приготування, потрібні інгредієнти, належність до кухні світу;
- у користувачів має бути можливість залишати відгуки на рецепти та приймати участь у формуванні рейтингу рецепту;
- у користувачів має бути можливість отримувати добірку рецептів, які вони мають спробувати приготувати;

- у користувачів має бути можливість бачити дані графіки із даними про вживані ними продукти та поживні речовини;
- має бути реалізований пошук рецепту за усіма даними, які може містить рецепт;
- у користувачів має бути можливість отримувати рекомендації щодо приготування страви на основі даних про його фізичне самопочуття, алергічні реакції на продукти та фізіологічні потреби у речовинах.

Таблиця 1.1

Результати аналізу існуючих рішень

Критерії / Рішення	Наявність бази рецептів, які готові до приготування	Наявність персоналізованих рекомендацій	Наявність візуалізації статистики користувача	Наявність можливості залишати відгуки до рецептів	Наявність пошуку рецептів за критеріями
Лікар-дієтолог	–	+	–	–	–
Cara: Food, Mood, Poop Tracker	+	+	+	–	–
Rise Up + Recover	–	–	+	–	+
Веб-сайт «Еда»	+	–	–	+	+

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Вибір мови програмування для розроблення серверної частини

Проаналізувавши функціональні вимоги до розробленого ПЗ, зроблено висновок, що оптимальною архітектурою в даному випадку виступає клієнт-серверна архітектура:

- модуль серверної частини відповідає за взаємодію з базою даних та зовнішніми ресурсами, агрегацію і збір даних рецептів;
- модуль клієнтської частини відповідає за реалізацію користувацького інтерфейсу.

Під час вибору мови програмування для реалізації серверної частини додатку було висунуто наступні вимоги:

- неблокуючий паралелізм виконання програми;
- жорстка типізація;
- бібліотеки для розробки серверних додатків;
- наявність активного товариства користувачів цієї мови;
- об’єктна орієнтованість.

Обрання мови програмування, яка б задовольняла цім вимогам, дозволить розробити прототип програмного забезпечення за мінімальний час.

2.1.1. *Python*

Python – високорівнева об'єктно-орієнтована мова програмування загального призначення із динамічною строгою типізацією [5]. На сьогоднішній день очолює список найпопулярніших мов програмування [6].

Основними з переваг мови Python виступають такі характеристики [7]:

- інтерпретованість. Текст програми не потрібно компілювати. Інтерпретатор мови програмування послідовно читає його та виконує інструкції. Це значно полегшує та прискорює процес відлагодження;

- кросплатформеність. Інтерпретатор мови доступний для більшості популятних платформ. Таким чином програмне забезпечення може працювати на таких популярних ОС як Windows, Linux, macOS тощо;
- об’єктно-орієнтовність. В основі програмування на Python лежить об’єктний підхід. Саме через це працювати в стилі ООП на даній мові програмування дуже зручно;
- інтегровність та розширюваність. Завдяки використанню сторонніх модулів з Python Package Index (PyPI) [8], Python може без зайвих зусиль інтегруватися з іншими платформами та мовами програмування;
- відкритість вихідного коду. Мова Python вільно поширюється та її сирцевий код зберігається у відкритому доступі;
- багата стандартна бібліотека. Бібліотека містить велику кількість модулів та функцій, які дозволяють полегшити розробку ПЗ;
- доступність великої кількості сервісів, середовищ розробки, та фреймворків;
- наявність великої кількості джерел інформації [5]. Python виступає однією з найпопулярніших мов та має велике співтовариство, тому можна легко знайти потрібну інформацію та відповіді на будь-які питання.

2.1.2. *C#*

C# – строго типізована об’єктно-орієнтована мова програмування, розроблена компанією Microsoft [9]. C# призначений для роботи з платформою .NET Microsoft. В свою чергу, .NET – це вільне, кросплатформене середовище з відкритим вихідним кодом, яка дозволяє створювати широкий спектр застосунків.

Основними з переваг мови C# виступають такі характеристики:

- об’єктна орієнтованість. Мова підтримує інструменти для реалізації парадигм ООП;
- використання статичної типізації. Реалізована перевірка відповідності типів, яка мінімізує кількість помилок, повязаних з перетворенням типів;
- багата стандартна бібліотека. величезна бібліотека вбудованих модулів та компонентів;
- наявний збирач сміття (garbage collector);
- велика спільнота розробників та наявність великої кількості джерел інформації.

Платформа ASP.NET Core являє собою технологію від компании Microsoft, яка призначена для створення веб-додатків. ASP.NET Core є продовженням ASP.NET та може бути запущена не лише на Windows, а і на Unix-подібних операційних систем: macOS, Linux [10].

2.1.3. *PHP*

PHP – популярна скриптова мова програмування з відкритим вихідним кодом [11]. На сьогоднішній день PHP входить у топ п’ять мов програмування для створення динамічних веб-додатків та веб-сайтів [12].
Основні переваги мови PHP:

- орієнтація на веб-розробку. PHP створювався, розвивався і підтримується як мова для створення веб-додатків. Багато конструкцій і рішень, створені для зручності роботи в веб-середовищі;
- гнучкість сфери застосування. При належному рівні володіння, за допомогою шаблонізатора можна створювати не тільки сценарії для веб-додатків, але й повноцінні програми;
- низький поріг входження. Вивчення PHP не вимагає багато часу. Вона є надзвичайною простотою для новачків;

- кросплатформеність. PHP може бути запущений в будь-якій операційній системі, включаючи unix-подібні системи;
- підтримка веб-серверів. Переважна більшість веб-серверів працюють з PHP;
- наявність великої кількості джерел інформації. PHP дуже популярна мова програмування, яку активно використовують починаючи з 1995 року. Більше ніж 73% існуючих веб-серверів були написані саме на цій мові програмування. Тому не дивно, що можна легко знайти потрібну інформацію яка стосується PHP.

Основні недоліки мови PHP:

- непослідовний синтаксис;
- протиріччя в стандартній бібліотеці. Код в стандартній бібліотеці переповнений різними залишками з різних мов, таких як C та Java;
- складність відловлювання помилок.

2.1.4. *JavaScript*

JavaScript – мова програмування, що є прототипно-орієнтованою, підтримує об'єктно-орієнтований, імперативний і функціональний стилі [13]. Дана мова програмування відповідає стандарту ECMAScript. JavaScript є універсальною мовою, яку можна використовувати і для створення динамічного клієнтського інтерфейсу у веб-браузері, і для створення асинхронних веб серверів за допомогою фреймворку Node.js.

До основних переваг мови JavaScript можна віднести:

- інтерпретованість. Вихідний програмний код інтерпретується, що дозволяє запускати його у будь-якому апаратному оточені, що підтримує інтерпретатор мови JavaScript;
- об’єктно-орієнтованість. ООП реалізується через прототипе наслідування. Дана мова реалізує усього чотири типи об’єктів:

об'єкти браузера, вбудовані об'єкти, об'єкти користувача та об'єкти документа;

- асинхронну модель виконання коду. Асинхронна модель виконання коду в JS реалізується за допомогою відповідних двигунів браузера, таких як SpiderMonkey в Mozilla Firefox та V8 у Google Chrome. та Ці двигуни будують асинхронну чергу подій, які обробляються механізмом, який називається «цикл подій» (event loop).

Недоліками JavaScript прийнято вважати:

- відсутність типізації даних. Поки виконання коду не дійде до потрібного рядка, важко зрозуміти чи працює вона коректо. Якби компілятор знав з якими типами даних він працює, то він міг би значно полегшити процес написання та відлагодження коду;
- низьку продуктивність. Оскільки JavaScript є інтерпретованою мовою програмування, вона однозначно програє в швидкості компільованим мовам.

2.1.5. *Висновки*

В даному підрозділі проаналізовано популярні на сьогоднішній день мови програмування. Досліджено технології та засоби розроблення серверної частини веб-додатків. Результати проведеного аналізу наведено в табл. 2.1.

На основі отриманих даних після аналізу чотирьох мов програмування в якості основної мови для реалізації серверної частини ПЗ було обрано мову C#, оскільки вона задовольняє всім вимогам, які було висунуто до неї, та не має очевидних недоліків у порівнянні із іншими розглянутими мовами.

Таблиця 2.1

Порівняння мов програмування для розроблення серверної частини

	Python	C#	PHP	JavaScript
Неблокуючий паралелізм виконання програми	+	+	+	+
Жорстка типізація	+/–	+	–	–
Об’єктно-орієнтованість	+	+	–	+/–
Бібліотеки для розробки серверних додатків	+	+	+	+
Активна спільнота розробників	+	+	+	+

2.2. Вибір технології для розроблення клієнтської частини

До технології для розроблення клієнтської частини було висунуті такі вимоги:

- структурованість;
- реалізація MVC;
- вичерпна кількість стандартних модулів та/або бібліотек;
- компонентний підхід до архітектури.

2.2.1. *Angular*

Angular – фреймворк для побудови веб інтерфейсів з відкритим вихідним кодом розроблений копманією Google [14].

Основні характеристики фреймворку Angular:

- model-View-Controller (MVC). Angular використовує в якості основного архітектурного патерну Model-View-Controller (MVC). Завдяки цій структурі Angular може розбивати завдання на логічні порції (chunks), що дозволяє зменшити час початкового завантаження веб-сторінки. Модель MVC також дозволяє розділити задачі, оскільки частина представлення View буде присутня на стороні клієнта, що значно скоротить кількість запитів у фоновому режимі. Крім того, зв'язок з цим інструментом працює в асинхронному режимі, що означає меншу кількість звернень до сервера;
- typeScript. Angular написаний з використанням мови TypeScript, яка є надбудовою над JavaScript. TypeScript повністю компілюється в JavaScript та зазвичай допомагає виявити та усунути поширені помилки під час написання коду [15];
- двостороння прив'язка даних. Angular реалізує механізм, за допомогою якого зміна значення в компоненті призводить до моментальної зміни значення в шаблоні, і навпаки;
- компонентний підхід.

Основні мінуси фреймворку Angular:

- відносно повільна продуктивність. Angular використовує брудну перевірку (dirty-checking), при двосторонній прив'язці даних, через що страждає продуктивність. Розробник потребує глибоких знань Angular, що б уникнути проблем пов’язаних з брудною перевіркою.
- різноманітність різних структур (Injectables, Components, Pipes, Modules тощо), що ускладнює вивчання Angular в порівнянні з React та Vue.js, які використовують єдиний «Component»;

- відсутність деталей у CLI документації. Інформація яку надає офіційна документація на GitHub. Розробникам доводиться витрачати більше часу на знаходження потрібної інформації.

2.2.2. *React*

React – JavaScript-бібліотека для розроблення користувацького інтерфейсу, основна мета якої – надання високої швидкості, простоти і масштабування при розробці односторінкових та мобільних додатки [16].

Основні характеристики фреймворку React:

- використання віртуального DOM. При зміні елементу на веб-сторінці зміни спочатку вносяться до віртуального DOM. Новий стан віртуального DOM порівнюється з поточним станом. У разі знаходження відмінності між станами React оновлює реальне DOM-дерево до поточного стану, з мінімальною кількістю маніпуляцій. Такий механізм взаємодії з елементами веб-сторінки працює набагато швидше та ефективніше;
- підтримка збірки у bundle та tree-shaking. Збірка (або бандлінг) - це процес виявлення імпортованих файлів і об’єднання їх в один «зібраний» файл (англ. «bundle»). Цей бандл після підключення на веб-сторінку завантажує весь додаток за один раз. Tree-shaking – це метод оптимізації бібліотек шляхом видалення коду з остаточного файлу, який фактично не використовується. Ці механізми використовуються для мінімізації завантаження ресурсів кінцевого користувача;
- візуалізація на стороні сервера (SSR). SSR - рендерінг на серверах клієнтської частини або універсального додатку в HTML;
- використання мови JSX. JSX представляє синтаксичний цукор мови JavaScript. Синтаксис JSX подібний до XML та HTML, та використовується для швидкого прототипування динамічних веб сторінок з використанням технології React.js.

Основні мінуси фреймворку React:

- відсутність реалізації MVC. У разі необхідності реалізації стану та моделі потрібно буде використовувати додаткові бібліотеки;
- погана документація. Завдяки постійним оновленням та усім створеним бібліотекам, React розвивається настільки швидко, що не вистачає часу для написання належної документації та уроків;
- часті оновлення. React постійно змінюється, і розробники повинні регулярно бути в курсі останніх новин.

2.2.3. *Vue.js*

Vue.js – JavaScript-фреймворк для побудови веб-інтерфейсів з відкритим вихідним кодом [17]. Найновіший серед розглянутих фреймворків, який стрімко розвивається.

Основні характеристики фреймворку Vue.js:

- використання віртуального DOM;
- продуктивність. Vue є надзвичайно швидким інструментом;
- простота використання. На відміну від інших фреймворкив Vue простий у навчанні, що робить його привабливим як для початківців, так і для давніх професіоналів;
- чудова документація;
- проста інтеграція проєкту. Дана особливість дозволяє миттєво почати використовувати Vue у своєму проєкті.

Основні мінуси фреймворку React:

- мала спільнота. Vue являється відносно новим фреймворком, Vue ще потребує часу, щоб розширити свою спільноту до розміру фанатів React або Angular;
- погана інтеграція із статично типізованою мовою програмування TypeScript [18].

2.2.4. *Висновки*

Провівши аналіз та порівнявши основні технології розроблення користувацького інтерфейсу було обрано React.js. Вирішальними факторами вибору фреймворку стали легкість у вивчені та швидка продуктивність порівняно з конкуруючими Angular та Vue.js.

Таблиця 2.2

Порівняння технологій для розроблення клієнтської частини

	Angular	React	Vue.js
Структурованість	+	+	+
Реалізація MVC	+	–	–
Вичерпна кількість стандартних модулів та/або бібліотек	+	+	+/–
Компонентний підхід до архітектури	+	+	+

2.3. Вибір СУБД

До СУБД для побудови серверної частини системи було висунуті такі вимоги:

- простота використання;
- безкоштовність;
- підтримка реляційної структури даних;
- відповідність ACID вимогам;
- підтримка повнотекстового пошуку;
- можливість розгортання на віддаленому сервері.

2.3.1. *MySQL*

MySQL – вільна реляційна система управління базами даних. На сьогоднішній день це одна з найпопулярніших СУБД [19]. MySQL може бути запущена на усіх основних ОС. Розробку та підтримку MySQL здійснює корпорація Oracle.

До переваг MySQL можна віднести такі характеристики як:

- простота та легкість у використанні. MySQL без зайвих проблем інсталюється на будь-яку операційну систему. Існую багато плагінів та додатків які спрощують роботу з базою даних;
- багатий функціонал. Система MySQL надає практично всі необхідні інструменти, які можуть знадобитися в реалізації практично будь-якого проєкту;
- безпека. MySQL містить багато вбудованих механізмів захисту та резервного копіювання інформації;
- масштабованість. MySQL може бути використана для роботи як і з великими об'ємами, так і з малими об'ємами даних;
- швидкість. Швидкодія системи забезпечується за рахунок спрощення використовуваних в ній стандартів.

До недоліків MySQL можна віднести:

- недостатню надійність. Показники надійності MySQL поступається деяким іншим популярним СУБД;
- низьку швидкість розроблення ПЗ. MySQL бракує технічної досконалості, що позначається на ефективності процесів розроблення ПЗ;
- повнотекстовий пошук. Деякі двигуни MySQL не підтримують повнотекстовий пошук.

2.3.2. *PostgreSQL*

PostgreSQL – реляційна СУБД з відкритим вихідним кодом [20]. Дана СУБД підтримує переважну більшість можливостей SQL:2011 [21]. Може бути запущена на усіх основних ОС.

До переваг PostgreSQL можна віднести:

- СУБД з відкритим вихідним кодом. PostgreSQL являє собою безкоштовне ПЗ з відкритим вихідним кодом;
- велику спільноту розробників;
- велику кількість доповнень. Розроблено велика кількість плагінів та доповнень, що поліпшують роботу з даною СУБД;
- масштабованість. PostrgreSQL реалізує можливість розширення функціоналу. Це досягається за рахунок збереження процедур;
- об'єктну орієнтованість. PostrgreSQL об'єктно-орієнтована СУБД, яка підтримкою успадкування.

До недоліків PostgreSQL можна віднести:

- ефективність використання ресурсів центральним процесором. PostgreSQL демонструє гірші показники продуктивності в порівнянні з конкуруючою MySQL при звичайних операціях читання;
- популярність. Популярністю ця СУБД похвалитися не може, хоча і має досить велику спільноту;
- хостинг. В силу названих вище проблем, іноді доволі складно знайти хостинг з підтримкою цієї СУБД.

2.3.3. *Висновки*

В даному підрозділі було проаналізовано існуючі реляційні безкоштовні СУБД. Результати аналізу наведено в табл. 2.3. З них видно, що обидві розглянуті СУБД відповідають висунутим умовам. Однак PostgreSQL надає порівняно більші можлівості повнотекстового пошуку.

Тому вона і була обрана для розроблення ПЗ в рамках даного дипломного проєкту.

Таблиця 2.3

Порівняння СУБД

	MySQL	PostgreSQL
Простота використання	+	+
Безкоштовність	+	+
Підтримка реляційної структури даних	+	+
Відповідність вимогам ACID	+	+
Можливість розгортання на віддаленому сервері	+	+
Підтримка повнотекстового пошуку.	–	+

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ПЗ

Перед початком будь-якого розроблення необхідно правильно організувати структуру програмного забезпечення. Ретельно спланована структурно-алгоритмічна організація є запорукою не лише швидкості розроблення та майбутньої можливості розширення системи, а також, що є дуже важливим для веб-додатків, швидкості роботи додатку.

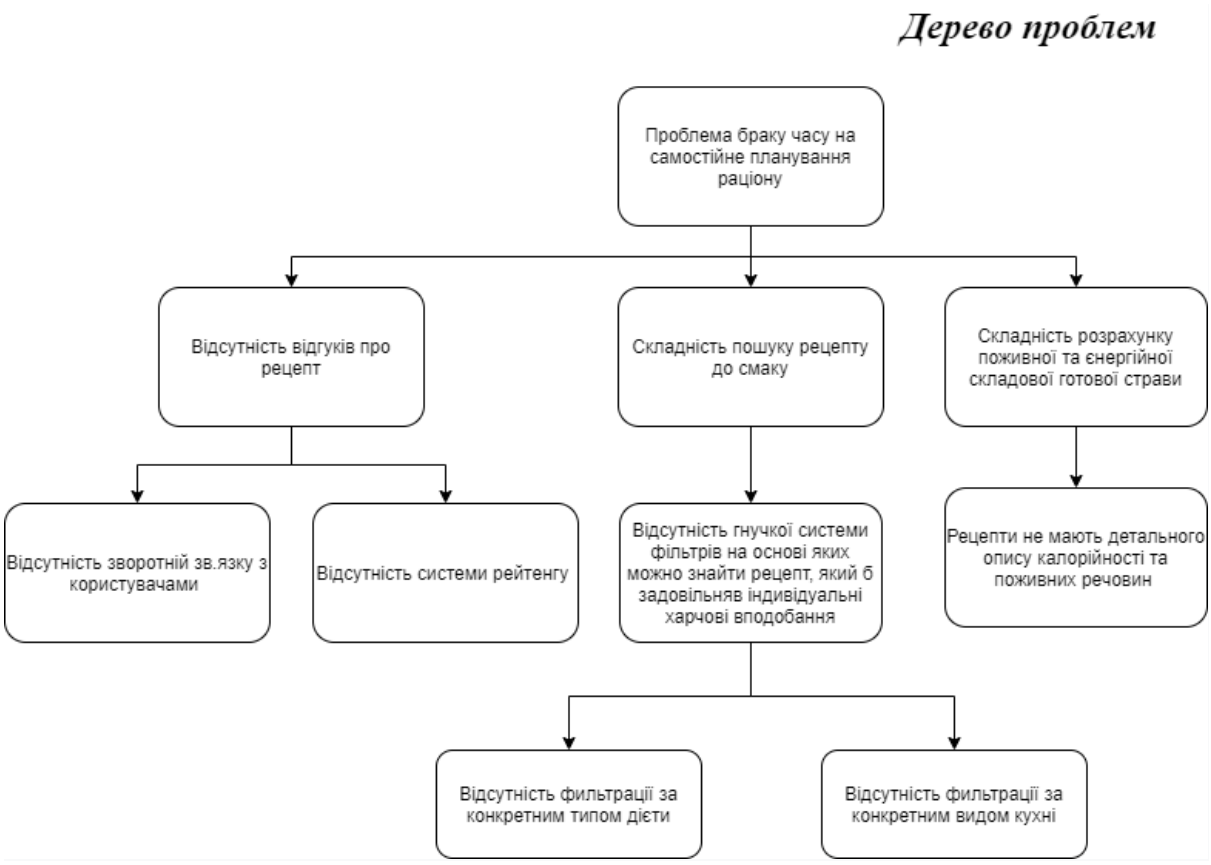


Рис. 3.1. Ілюстрація структури дерева проблем, які вирішує розроблюване програмне забезпечення

3.1. Вимоги до розроблюваного ПЗ

3.1.1. *Аналіз проблематики та цілей формування раціону користувача*

Головною проблемою формування раціону виступає брак часу на самостійне планування цього раціону. Дану проблему можна розкласти на наступні, більш конкретні проблеми:

- складність розрахунку поживної та енергетичної складової готової страви. Розрахунок калорійності та вмісту органічних сполук є основним у більшості відомих дієтах. Проте, переважна більшість ресурсів не надає детального опису калорійності та поживних речовин готової страви та її інгредієнтів. Через це користувач повинен власноруч шукати дану інформацію і робити розрахунки;
- складність пошуку рецептів до смаку. У кожного з нас є особисті вподобання в їжі. Через відсутність гнучкої системи фільтрів (таких як тип дієти або вид кухні), користувач вимушений або перераховувати всі потрібні йому нюанси у пошуковій строчці, або ж перебирати очима всі рецепти у пошуку найкращого;
- відсутність відгуків про рецепт. Система рейтингу відіграє важливу роль при виборі будь-чого. Реальні відгуки людей про рецепт можуть бути корисні при формуванні раціону. Введення системи зворотного зв’язку дозволить оптимізувати алгоритм рекомендацій та підбору рецептів.

Вище перераховані проблеми можна візуалізувати за допомогою дерева проблем, яке представлено на рис. 3.1.

Провівши аналіз дерева проблем, було висунуто цілі, що мають бути досягнуті при розробленні веб-додатку. Загальною метою даного програмного забезпечення є надання користувачу можливості без зайвих труднощів підібрати індивідуальний раціон.

На основі отриманих цілей було сформовано дерево результатів, що мають бути отримані в процесі розроблення продукту(рис. 3.3).

Отже, в результаті аналізу проблематики, цілей та результатів було зібрано вимоги, які висуваються до розроблюваного програмного забезпечення.

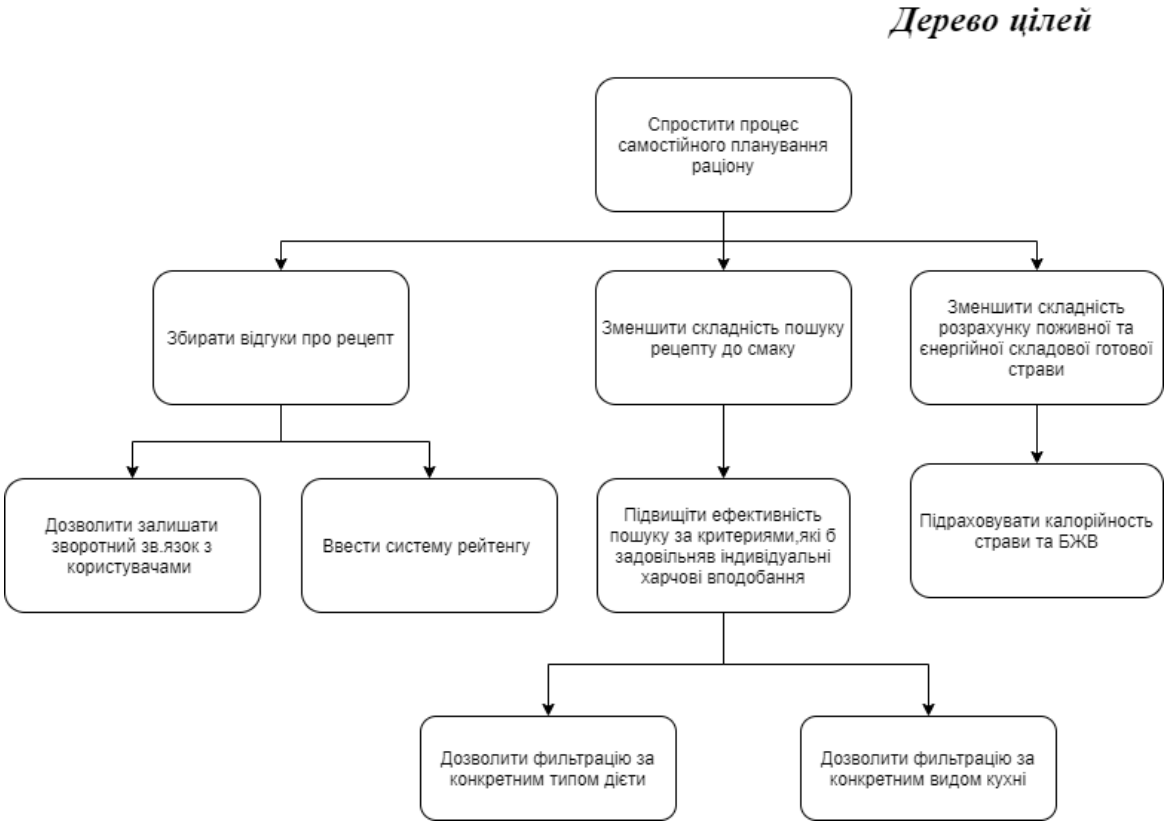


Рис. 3.2. Ілюстрація структури дерева цілей продукту для розроблюваного програмного забезпечення

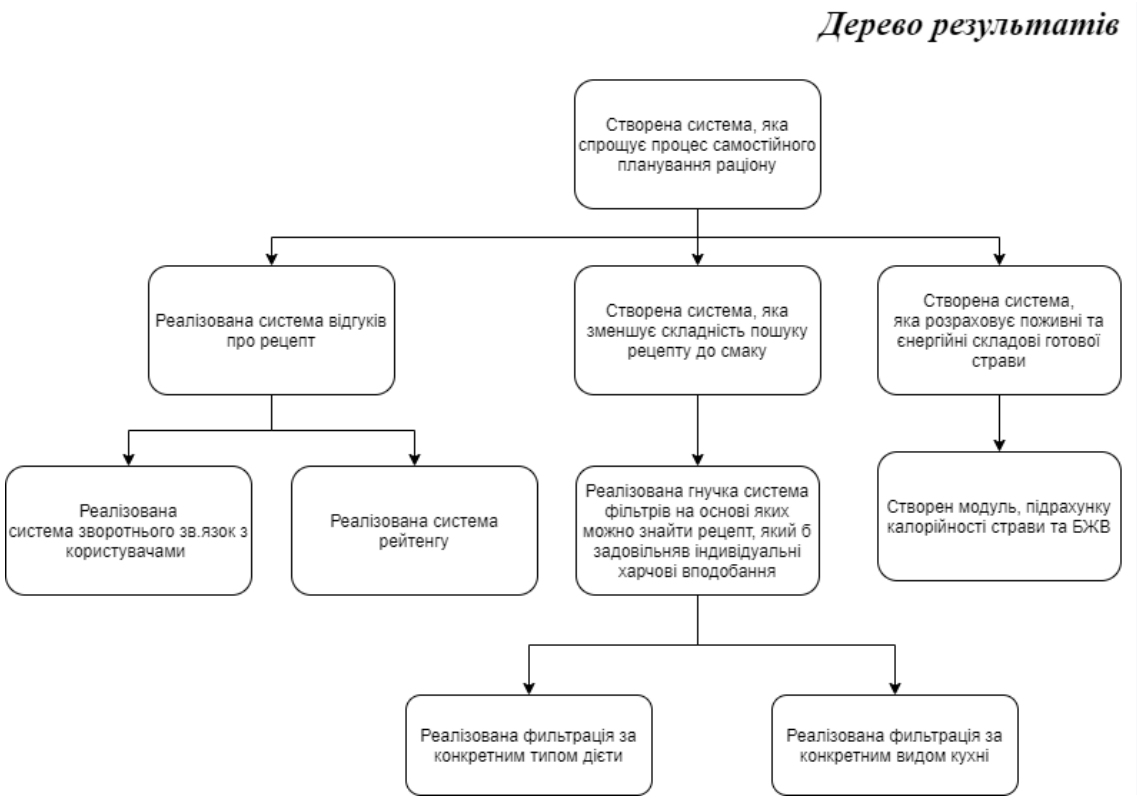


Рис. 3.3. Дерево результатів продукту

3.1.2. *Загальний опис системи*

Веб-додаток для підтримки формування раціону користувача на основі платформи AWS – це додаток, який складається з двох частин: клієнтської (front end) та серверної (back end). Кожна з частин виконує свою функцію та має свої функціональні та нефункціональні вимоги.

Основна задача даного програмного забезпечення полягає в тому, щоб даті користувачу можливість без зайвих труднощів підібрати індивідуальний раціон.

Даний веб-додаток підтримує 3 ролі користувачів:

- авторизований користувач. Дана роль користувача має змогу ввести бажану денну норму калорій и БЖВ та розподілити їх на декілька приймів їжі. Після того, як всі необхідні параметри будуть прийняті, система буде підбирати доцільні рецепти та пропонувати їх користувачу. Також користувач може самостійно обрати рецепт з каталогу. У цьому йому допоможе зручна система фільтрів. Калорійність и кількість БЖВ будуть враховані у денну норму. Окрім цього, авторизований користувач може залишати зворотній зв’язок: лишати відгук та оцінку рецепту по декільком категоріям;
- неавторизований користувач. Даному типу користувача доступний лише перегляд всіх рецептів представлених на сайти;
- адміністратор. Дана роль має повний контроль управління всім контентом на сайті: видалення та редагування рецептів, перегляд інформації про користувачів, блокування та розблокування користувачів, видалення відгуків.

Як тільки користувач переходить на сторінку, він має роль неавторизованого користувача. Він має змогу лише переглядати всі рецепти представлені на сайті, механізм підбору раціону та відгуки про рецепт будуть недоступні. Для того щоб користувачу став доступним весь функціонал додатку, він має зареєструватися у системі. Після реєстрації

йому буде надана можливість налаштувати особисті дані, які включають бажану денну норму калорій та БЖВ (користувач має змогу корегувати особисту інформацію у будь-який час). Після внесення потрібної інформації активується механізм підбору раціону, який буде пропонувати користувачу нові рецепти, які задовольняли б його побажання.

3.1.3. *Вимоги до веб-додатку*

Проаналізувавши предметну галузь, було сформульовано такі функціональні вимоги до розроблюваного ПЗ:

1. Система має надавати можливість реєстрації нових користувачів та автентифікації існуючих користувачів.
2. Система має визначати приблизну денну кількість калорій та БЖВ, на основі параметрів користувача.
3. Система має дозволити перегляд доступних рецептів на сайті.
4. Система має надати гнучку систему пошуку рецептів за наступними параметрами:
    4.1. Категорія страви: салат, суп, десерти та т.п.
    4.2. Тип кухні: італьська, французька, українська та т.п.
    4.3. Тип меню: вегетаріанське, веганське, меню при діабеті та т.д.
    4.4. Рейтинг страви.
    4.5. Інгредієнти: включати, або ж навпаки, виключати певні інгредієнти.
    4.6. Час приготування.
    4.7. Складність приготування.
    4.8. Калорійність страви.
    4.9. Вміст БЖВ.
5. Система має рекомендувати декілька рецептів для користувача на основі денної норми організму в калоріях та БЖВ.

6. Система має дозволити зберігати рецепт, що сподобався користувачу, до списку вподобань.
7. Система має вираховувати калорійність обраного рецепту при підрахунку залишків від денної норми споживання калорій та БЖВ.
8. Система має надавати можливість авторизованим користувачам залишати зворотній зв’язок:
   8.1. Залишати відгуки та прикріпляти фото приготованої власноруч страви.
   8.2. Залишати оцінку: загальну оцінку рецепту та оцінку складності.
9. Система має надавати адміністратору можливість додавати/видаляти/модифікувати рецепти.
10. Система має надавати адміністратору право на перегляд/модифікації/видалення користувачів.

Для кращого розуміння проілюструємо сформовані функціональні вимоги за допомогою відповідних діаграм прецедентів (англ. «use case diagram»).

Прецедент 1: «Зареєструватися та/або авторизуватися у системі» – описує процес початку взаємодії із системою (рис. 3.4).

1. Користувач відкриває додаток.
2. Якщо користувач не має облікового запису в системі, він може пройти процедуру реєстрації.
3. Користувач переходить на форму реєстрації
4. Користувач вводить логін, електронну адресу та пароль.
5. Якщо користувач вже був зареєстрований:
   5.1. Користувач переходить на сторінку авторизації.
   5.2. Користувач вводить логін або електронну адресу та пароль.
6. Якщо користувач забув пароль від свого облікового запису:
   6.1. Користувач вводить запит на відновлення паролю.

6.2. На прив’язану до облікового запису електронну адресу прийде лист-підтвердження зміни паролю з посиланням на форму відновлення.

7. Користувач переходить на форму відновлення та генерує новий пароль, який надалі буде використовуватися для входу на сайт.

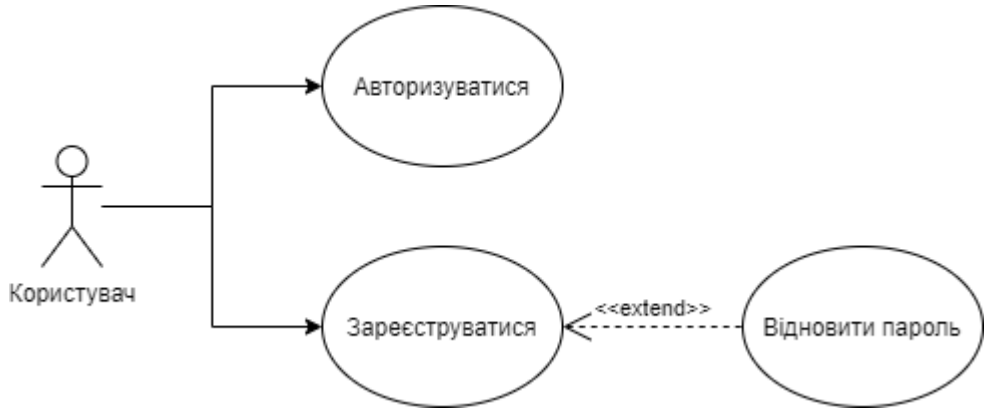


Рис. 3.4. Прецедент «Зареєструватися та/або авторизуватися у системі»

Прецедент 2: «Визначити та/або скорегувати індивідуальну денну норму споживання БЖВ» (рис. 3.5).

1. Користувач вводить параметри для розрахунку денної норми: стать, вік, зріст та вага.
2. Система розраховує денну норму користувача за формулою Харриса-Бенедикта та зберігає результат.
3. Якщо в користувача з’являється потреба скорегувати кількість калорій та/або БЖВ, він може їх змінити у особистому кабінеті.

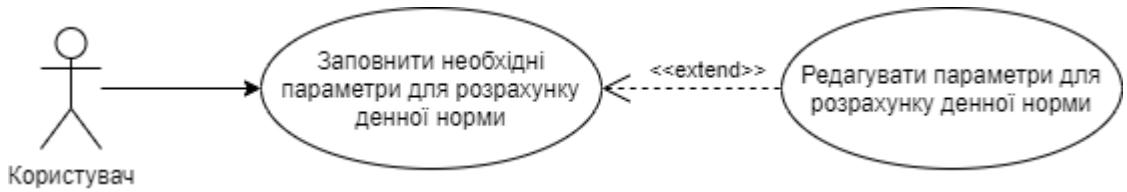


Рис. 3.5. Прецедент «Визначити та/або скорегувати індивідуальну денну норму споживання БЖВ»

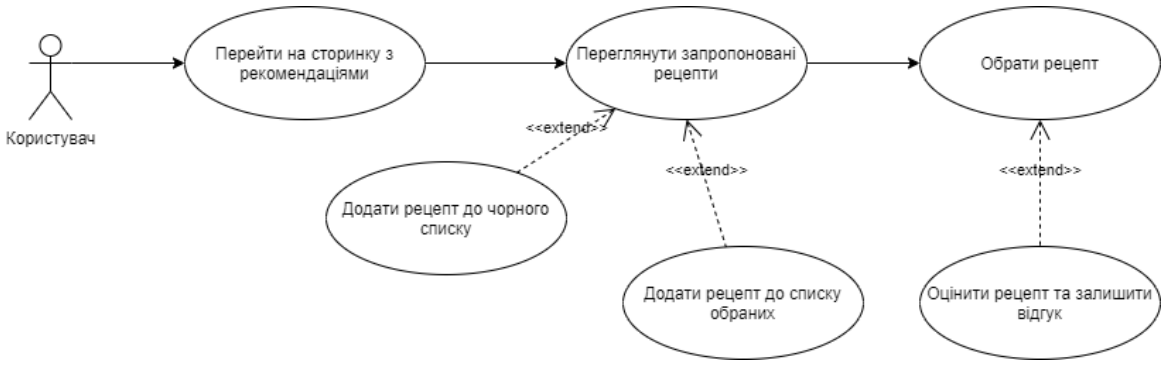


Рис. 3.6. Прецедент «Отримати рекомендацію рецепту»

Прецедент 3: «Отримати рекомендацію рецепту» (рис. 3.6).

1. Користувач нажимає на кнопку початку пошуку рецептів, після чого система його перенаправляє на сторінку з рекомендаціями.
2. Користувачу буде рекомендовано 3 різних страви.
   2.1. Якщо користувач вказав у своєму профілі значення «Тип меню», рецепти рекомендації будуть відповідати обраному меню.
   2.2. Якщо користувач не вказав значення «Тип меню», то рекомендації рецептів будуть братись з різних типів меню.
3. Якщо користувач побачив рецепт, який йому не сподобався і який він не хоче більше бачити в своїх рекомендаціях, то він може натиснути на кнопку «Не рекомендувати більше». Даний рецепт більше не буде відображатися в рекомендаціях, проте його можна буде знайти у глобальному каталозі рецептів, та поновити натиснувши на кнопку «Показувати у рекомендованих».
4. Якщо користувачу сподобався рецепт, але в даний момент він не буде його обирати, то він може натиснути на кнопку «Зберегти». Даний рецепт внесеться в лист збережених рецептів.
5. Користувач обирає один з запропонованих рецептів.
6. Кількість калорій та БЖВ буде вирахувана в денної норми.

7. Система запропонує оцінити рецепт та залишити відгук.

Прецедент 4: «Залишити зворотній зв’язок до рецепту» (рис. 3.7).

1. Користувач обирає рецепт.
2. Користувач вводить текст у відповідну форму для відгуків.
3. Користувач оцінює загальні враження до рецепту.
4. Користувач оцінює складність рецепту.
5. Користувач може прикріпити фото страви, яку він приготував.

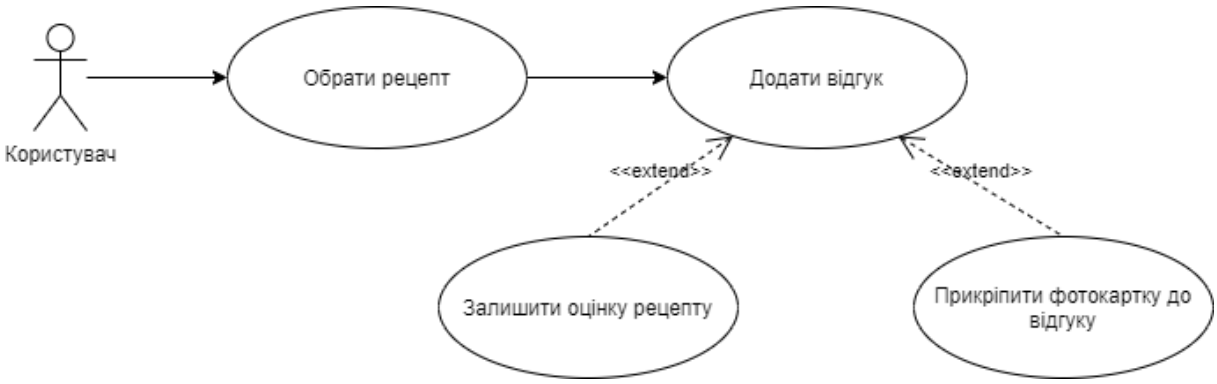


Рис. 3.7. Прецедент «Отримати рекомендацію рецепту»

Прецедент 5: «Залишити зворотній зв’язок до рецепту» (рис. 3.8).

1. Користувач переходить на сторінку каталогу рецептів.
2. Користувач налаштовує фільтри.
3. Користувач натискає кнопку «Пошук».
4. Користувач обирає рецепт з результатів пошуку.
5. Наступні кроки коректні лише для авторизованого користувача.
6. Якщо калорійність страви перевищує залишок калорій денної норми, система нотифікує користувача про перебільшення денної норми.
7. Користувач може відмітити рецепт як «Приготований».
8. Калорійність та БЖВ віднімаються від їх залишку на день.

Рис. 3.8. Прецедент «Залишити зворотній зв’язок до рецепту»

Прецедент 6: «Взаємодія зі списком збережених рецептів» (рис. 3.9).

1. Користувач переходить до списку збережених рецептів у пункті меню.
2. Користувач переглядає список збережених рецептів.
3. Користувач може перейти на опис конкретного рецепту.
4. Користувач може видалити рецепт зі списку.
5. Користувач може додати новий рецепт до списку.

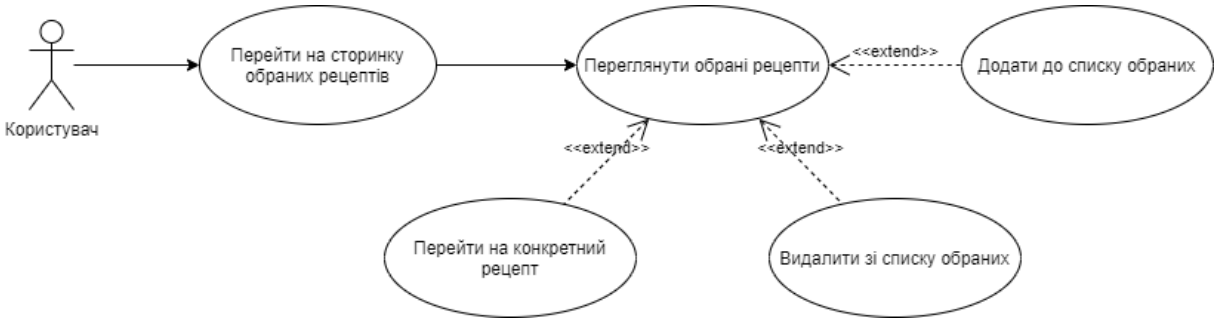


Рис. 3.9. Прецедент «Взаємодія зі списком збережених рецептів»

Прецедент 7: «Взаємодія зі списком заблокованих рецептів» (рис. 3.10).

1. Користувач переходить до списку заблокованих рецептів у пункті меню.
2. Користувач переглядає список заблокованих рецептів.
3. Користувач може перейти на конкретний рецепт.
4. Користувач може видалити рецепт зі списку. Надалі рецепт може бути рекомендованим для користувача.

5. Користувач може додати новий рецепт до списку. Надалі система не буде рекомендувати рецепт для користувача.

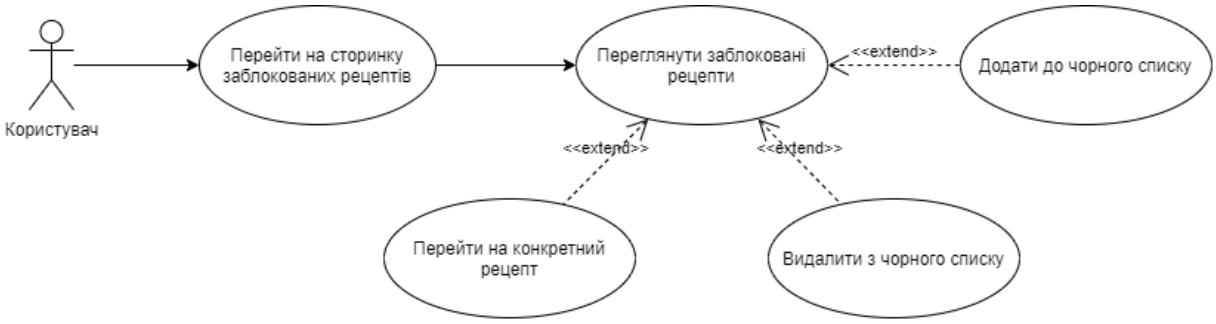


Рис. 3.10. Прецедент «Взаємодія зі списком заблокованих рецептів»

Розглянемо прецеденти які доступні лише адміністратору системи.

Прецедент 8: «Взаємодія зі списком користувачів системи» (рис. 3.11).

1. Адміністратор переходить до списку всіх користувачів даного додатку.
2. Адміністратор переглядає список всіх користувачів.
3. Адміністратор може переглянути профіль конкретного користувача.
4. Адміністратор може заблокувати користувача.
5. Адміністратор може розблокувати користувача.

Рис. 3.11. Прецедент «Взаємодія зі списком користувачів системи»

Прецедент 9: «Взаємодія зі списком рецептів» (рис. 3.12).

1. Адміністратор переходить до списку рецептів.

2. Адміністратор переглядає список всіх рецептів.
3. Адміністратор перейти на конкретний рецепт.
4. Адміністратор може створити новий рецепт.
5. Адміністратор може видалити існуючий рецепт.
6. Адміністратор може вносити зміни в рецепт.

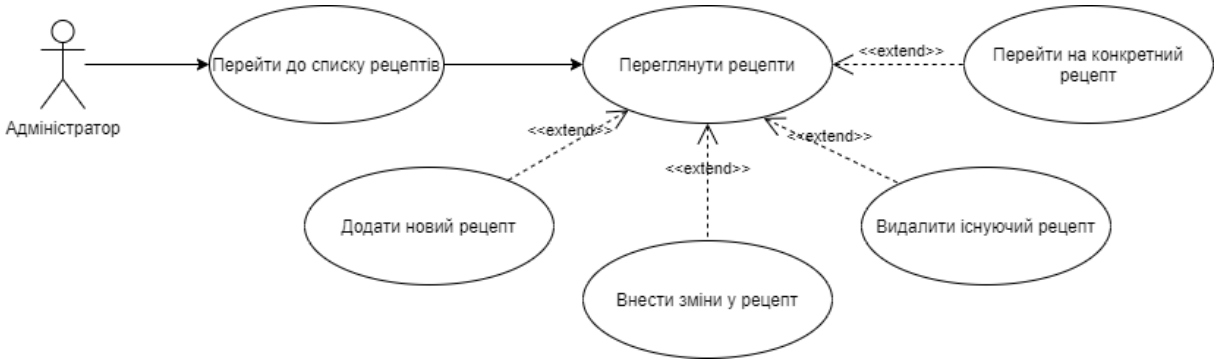


Рис. 3.12. Прецедент «Взаємодія зі списком рецептів»

Прецедент 10: «Взаємодія з відгуками до рецепту» (рис. 3.13).

1. Адміністратор переходить до конкретного рецепту.
2. Адміністратор переглядає відгуки які прикріплені до даного рецепту.
3. Адміністратор може видалити відгук.
4. Адміністратор може змінювати відгук.

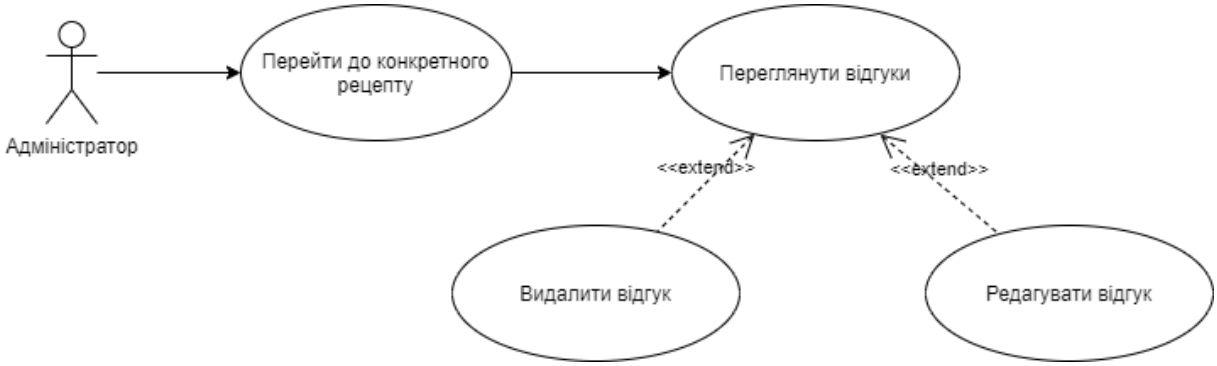


Рис. 3.13. Прецедент «Взаємодія з відгуками до рецепту»

Сформулюємо нефункціональні вимоги до розроблюваного ПЗ. Для більшої зручності згрупуємо вимоги по категоріях:

- вимоги до безпеки (табл. 3.1);
- вимоги до продуктивності (табл. 3.2);
- вимоги до реалізації (табл. 3.3).

Таблиця 3.1

Вимоги до безпеки

Код	Опис вимоги
SEC-1	Паролі користувачів повинні зберігатися у захешованому вигляді.
SEC-2	Система повинна перевіряти всі дані, які надходять від користувача.
SEC-3	Система повинна бути захищена від поширених видів атак (XSS, CSRF, SQL Injection).

Таблиця 3.2

Вимоги до продуктивності

Код	Опис вимоги
PRF-1	Веб-сторінка має ініціалізуватися менше ніж за 3 секунди після завантаження файлі в браузер користувача.
PRF-2	Система має відповідати на запити менш ніж за 1 секунду
PRF-3	Система має коректно працювати при навантажені в 1000 запитів на секунду.

3.2. Опис архітектури системи

Для реалізації даного веб-додатку було обрано клієнт-серверну архітектуру [22]. Вона передбачає розділення програмного застосунку на функціональні модулі: клієнтський інтрефейс та сервер. Клієнтський інтерфейс здебільшого реалізують у вигляді веб-сайту, який за допомогою

HTTP запитів отримує дані від веб-серверу та відображає їх користувачу. Веб-сервер в даному випадку представляє собою ПЗ, що запущене на віддаленому комп'ютері і яке відповідає за обробку, збереження та надання доступу до інформації. До переваг клієнт-серверної моделі відносять:

- слабку зв'язність інтерфейсу користувача та моделі зберігання даних;
- можливість залучення до частини обчислень локального ПК користувача;
- можливість перенесення ресурсномістких операцій на веб-сервер.

Таблиця 3.3

Вимоги до реалізації

Код вимоги	Опис вимоги
IMP-1	Веб-додаток має коректно працювати в оточенні сучасних браузерів, якими користується щонайменше 5% користувачів.
IMP-2	Для побудови стейт машини має використовуватися AWS Step Function
IMP-3	Система має підтримувати автоматичну доставку коду (CI) та розгортання (CD)
IMP-4	Серверна частина веб-додатку має бути побудована за мікросервісною архітектурою та розгорнута за допомогою AWS Lambda.

3.3. Опис архітектури модуля клієнтського інтерфейсу

Для реалізації клієнтської частини даного веб-додатку було використано архітектуру SPA (укр. «односторінковий додаток»). На відміну від розповсюдженої практики розроблення веб-сторінок, при якій для отримання кожної нової сторінки, браузеру користувача слід було робити запит до сервера та завантажувати новий документ, і лише після цього

відображати нову сторінку, SPA передбачає динамічну перебудову інтерфейсу веб-сторінки на основі даних, отриманих в результаті виконання HTTP-запиту за технологією AJAX [23] до веб-сервера, без її перезавантаження. Це дозволяє оптимізувати використання інтернет-трафіку та пришвидшити операції оновлення вмісту веб-сайту, тим самим не змушуючи користувача переривати його роботу, забезпечуючи йому кращий користувацький досвід.

Готове ПЗ клієнтського інтерфейсу представляє собою наступні файли:

- файл із базовою сторінкою HTML, яка підключає інші файли та є початковою точкою ініціалізації веб-додатку;
- файл із стилями веб-сторінки;
- файл із мініфікованими скриптами на мові JavaScript.

Ці файли було розміщено на CDN [24] (укр. «мережа доставки контенту») для забезпечення їх високої доступності та зменшення часу відклику. До посилання на базову сторінку HTML було прив’язано домене ім’я додатку. В результаті було отримано працюючу веб-сторінку, доступну до використання за посиланням на неї всім користувачам мережі Інтернет.

3.4. Опис архітектури модуля веб-серверу

До серверного ПЗ висуваються наступні вимоги:

- висока доступність – дані мають віддаватися клієнту із мінімальною затримкою;
- висока надійність – в разі виходу із ладу одного із веб-серверів, жоден користувач не має відчувати проблем із користуванням веб-додатком;
- горизонтальна масштабованість – можливість реагувати на зростання навантаження на сервер шляхом збільшення кількості екземплярів програми, замість збільшення обчислювальних ресурсів кожного з них.

Реалізація веб-додатку для підтримки формування раціону користувача передбачає використання платформи хмарних обчислень AWS.

Загальноприйнятими методиками проєктування ПЗ є його розроблення на основі монолітної чи мікросервісної архітектур. До переваг першої відносяться: висока швидкість передачі даних між компонентами, структурованість, зазвичай – використання єдиної кодової бази. Недоліками вважаються висока зв'язність компонентів, погана масштабованість, високі вимоги до апаратних ресурсів, можливість використовувати лише одну мову програмування. Мікросервісна архітектура, навпроти, дозволяє розробляти гнучкі системи, компоненти яких слабко пов'язані один з одним, добре масштабуються, дозволяють використовувати декілька мов програмування. Недоліками є складність проєктування та підтримки працездатності суцільної системи.

Для реалізації серверної частини було обрано саме мікро-сервісну архітектуру. Це дозволило використати сервіс платформи хмарних обчислень AWS, який працює за моделью Faas [25] AWS Lambda. Це дозволило запускати кожен екзмепляр мікросервісу в окремому програмному середовищі із підримкою багаторазового масштабовування, тим самим реалізуючи безсерверну модель проєктування ПЗ, коли роль бекенду відіграє один екземпляр лямбда-функції із запущеним в ньому мікросервісом.

3.4.1. *Особливості проєктування мікросервісної архітектури з використанням платформи хмарних обчислень AWS*

При розробленні веб-додатку для підтримки формування раціону користувача відповідно до теми дипломного проєкту обрано використання мікросервісної архітектури із застосуванням платформи хмарних функцій AWS для розгортання сервісів. Сформуємо перелік функціональних вимог, яким повинен відповідати розроблюваний веб-додаток. Це, зокрема, забезпечення отримання, оброблення та збереження інформації про

рецепти, робота із даними користувачів, реалізація алгоритму формування рекомендацій на основі даних про вподобання користувача. На основі даного переліку виділимо функціональні модулі, на які можна поділити веб-додаток і кожен з яких повинен підтримувати:

- безпосередню роботу із обмеженим переліком сутностей, які в ідеалі можна було б винести в окрему базу даних, доступ до якої є лише в цього модуля;
- інкапсуляцію внутрішньої реалізації за публічним інтерфейсом;
- можливість масштабувати мікросервіс для забезпечення стійкої роботи всієї системи;
- слабку зв'язаність із рештою мікросервісів.

На основі цих вимог було сформовано перелік модулів, на які було розділено розроблюваний веб-додаток (рис.3.14). В табл.3.4, описано отримані мікросервіси та їх зовнішні сутності.

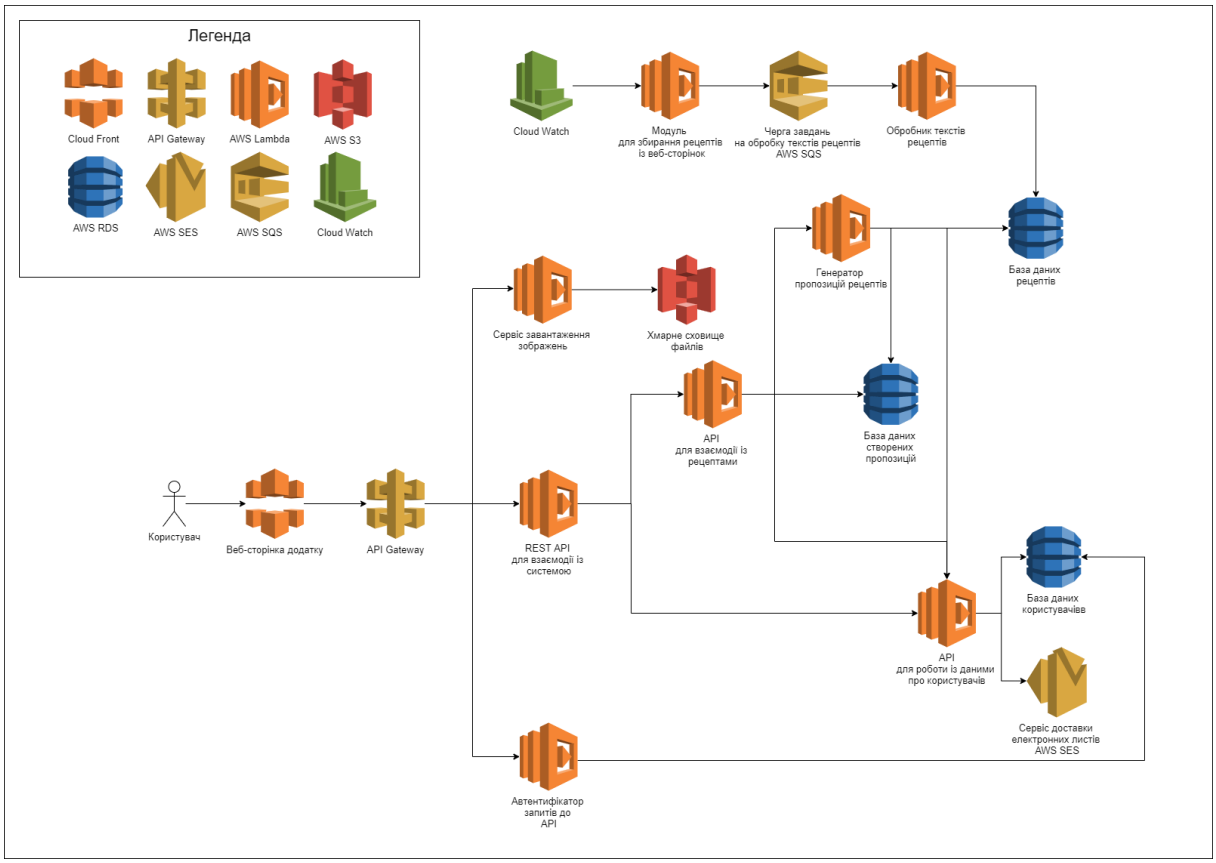


Рис. 3.14. Структурна схема серверної частини системи

Таблиця 3.4

Модулі серверної частини веб-додатку

Сервіс	Зовнішні залежності	Сутності
Модуль для збирання рецептів із веб-сторінок	Доступ до зовнішньої мережі Інтернет	Веб-сторінка
	Черга завдань на обробку текстів рецептів AWS SQS	
Обробник текстів рецептів	Черга завдань на обробку текстів рецептів AWS SQS	Рецепт
	База даних рецептів	
API для роботи із даними про користувачів	База даних користувачів	Користувач
	Сервіс доставки електронних листів AWS SES	
Генератор пропозицій рецептів	API для роботи із даними про користувачів	Користувач
	База даних рецептів	Рецепт
	База даних створених пропозицій	Пропозиція
	Система повнотекстового пошуку	
API для взаємодії із рецептами	Генератор пропозицій рецептів	Користувач
	База даних створених рецептів	Рецепт
	API для роботи із даними про користувачів	Пропозиція
Автентифікатор запитів до API	База даних користувачів	Користувач
REST API для взаємодії із системою	AWS ApiGateway	
	API для роботи із даними про користувачів	
	API для взаємодії із пропозиціями рецептів	

3.4.2. *Опис сервісу для збирання рецептів із веб-сторінок*

Сервіс для збирання рецептів із веб-сторінок представляє собою ПЗ, призначене для пошуку даних про рецепти на спеціалізованих тематичних веб-сайтах. Адреси веб-сайтів, на яких він має шукати рецепти, задаються через конфігурацію розробниками.

Збирач має відвідати всі сторінки веб-ресурсу та знайти на них посилання на сторінки із рецептами. При цьому сторінки можуть містити циклічні посилання, або бути вже оброблені та збережені до БД. Тому цей сервіс записує до БД всі адреси сторінок, які він відвідував та оброблював. Якщо поточна сторінка із рецептом вже була оброблена та збережена, при наступному запуску сервіс, вона буде пропущена. Зазвичай, нові рецепти на веб-сайтах з’являються не частіше кілька разів на день, тому доцільно запускати збирач раз на день. Це мінімізує марну трату ресурсів AWS Lambda.

Для того аби зменшити загальний час, потрібний для обробки всіх сторінок веб-сайту, було використано чергу завдань AWS SQS. В неї сервісом-збирачем додаються нові посилання, які слід обробити та з неї отримуються ще не оброблені. Завдяки використанню AWS Lambda як середовища для розгортки ПЗ сервісу для збирання даних рецептів, можна гнучко керувати кількістю одночасно працюючих екземплярів програми, не дозволяючи викликати блокування цільового веб-ресурсу через автоматичні запити до нього.

3.4.3. *Опис сервісу обробки рецептів*

Модуль для оброблення рецептів представляє собою програму, прив’язану до черги завдань AWS SQS, в яку сервіс збирання даних рецептів додає записи про знайдені ним сторінки із рецептом.

Його призначення – це обробка даних про рецепт, обчислення його поживності та складових, привласнення рецептам пошукових тегів. Для

цього він розбирає DOM-дерево сторінки і шукає в ньому потрібну йому інформацію. Як і в сервісі збору даних рецептів із веб-сайтів, екземпляри цього модуля працюють паралельно, а їх одночасна кількість регулюється через конфігурацію AWS Lambda.

3.4.4. *Опис сервісу API даних користувачів*

За надання інтерфейсу доступу до даних про користувачів системи та маніпуляцій із ними було створено сервіс API даних користувачів. До цих операцій відносяться:

- оновлення персональної інформації;
- оновлення даних для аутентифікації;
- отримання даних профілю;
- деактивація профілю;
- зміна вимірів тіла користувача;
- зміна денної норми калорій користувача;
- реєстрація нових користувачів.

Дані про користувачів зберігаються в БД. Цей сервіс виступає інтерфейсом, через який можна виконати дію над ними. Сам веб-сервіс побудовано за моделлю REST [26] (англ. «передача репрезентативного стану»). У загальному випадку REST є простим інтерфейсом управління інформацією, який виключає використання додаткових внутрішніх прошарків. Кожна сутність однозначно визначається глобальним ідентифікатором, таким як URL.

Кожній операції, яку можна застосувати до профілей користувачів, було присвоєно ідентифікатор. Кожному запиту до API сервісу ставиться у відповідність назва операції. За контроль можливості виконати запит також відповідає цей сервіс. Він перевіряє дотримання правил безпеки кожним запитом та в разі, якщо вони порушені, перериває його обробку і повертає помилку про порушення прав доступу.

3.4.5. *Генератор пропозицій рецептів*

Однією із основних функціональних можливостей веб-додатку для формування раціону користувача є створення пропозицій користувачам. Для цього було розроблено окремий сервіс, який виступає в ролі генератора таких пропозицій. Для цього він оперує даним про виміри тіла користувача, та його вподобання в їжі, відомості про алергени, тощо. В результаті він будує запит до БД із рецептами та шукає серед них декілька рецептів, які ще не пропонував користувачеві. Для швидкого пошуку серед рецептів, кожному з них було привласнено теги, за якими їх можна було б швидко відфільтрувати.

Запит до БД із рецептами може містити фільтрацію записів за такими критеріями:

- відсутність інгредієнтів, які користувач відмовився вживати;
- приналежність до улюблених користувачем видів кухень;
- мінімальний рейтинг рецепту;
- відповідність лімітам на БЖВ та калорії для користувача;
- тип блюда.

Користувач може самотужки ініціювати отримання рекомендації рецепту через веб-інтерфейс додатку.

3.4.6. *Сервіс API для взаємодії із рецептами*

Взаємодія із БД рецептів відбувається таким самим чином, як і з даними про профілі користувачів. За це відповідає виділений сервіс API для взаємодії із рецептами. Він оперує записами в БД і також реалізує REST інтерфейс для доступу до них.

До основних операцій, які підтримуються цим сервісом відносяться:

- пошук серед рецептів за встановленими критеріями;
- отримання рекомендацій рецепту для обраного користувача;
- залишення відгуку на рецепт.

3.4.7. *Автентифікатор запитів до API*

Оскільки деякі операції над даними в БД потребують перевірки авторизації користувача, слід було реалізувати механізм перевірки авторизації запитів від користувачів. Для цього було розроблено окремий сервіс, який дозволяє на основі даних, отриманих із заголовків HTTP запитів, отримувати ідентифікатор користувача, який їх надіслав.

В якості механізму авторизації користувача було використано технологію JWT [27] (англ. «JSON веб токен»). Принцип роботи JWT полягає в тому що веб-сервер віддає клієнту рядок, в якому закодовано деяку інформацію. В подальшому користувач може відправляти запити до серверу та прикріплювати цей рядок до них. Сервер ідентифікує відправника за тими даними, які було закодовано в строку. Слід зазначити, що дані не є захищеними від читання третіми особами, проте стандарт JWT гарантує, що вони не будуть скомпрометовані. Це досягається завдяки підпису даних, який обчислюється із використанням хеш-суми прикріплених до JWT даних та секретного ключа, яким володіє лише веб-сервер.

В якості даних, за якими сервіс авторизації ідентифікує користувача було використано його ідентифікатор БД. Його було закодовано в JWT та повернуто клієнту при виконанні ним операції входу в профіль.

3.4.8. *REST API для взаємодії із системою*

Для того аби приховати деталі реалізації серверної частини веб-додатку для формування раціону користувача, було побудовано фасад над API всіх сервісів засобами AWS API Gateway [28]. Метою застосування даного сервісу було описання RESTful [29] інтерфейсу системи у вигляді API із єдиною точкою доступу. Завдяки цьому стало можливим винести операції перевірки, кешування та перевірки авторизації на рівень вище, ніж усі мікросервіси – на рівень AWS API Gateway. Також слід зазначити, що

AWS не вимагає мінімального платежу за користування API Gateway, сума оплати вираховується як додаток кількості реально отриманих запитів, на вартість обробки одного запиту в рамках поточного тарифу.

3.5. Опис структур даних системи

Проаналізувавши висунуті вимоги до ПЗ, було спроєктовано архітектуру БД, яка включає такі сутності:

1. User – користувач системи.
    a. id – унікальний ідентифікатор користувача у форматі UUID, первинний ключ;
    b. name – ім’я користувача у системі;
    c. username – унікальне ім’я користувача для входу у систему;
    d. password – пароль користувача для входу у систему. Всі паролі зберігаються у захешованому вигляді. В якості алгоритма хешування було обрано SHA256;
    e. email – електронна адреса користувача. Може бути використана як унікальне ім’я для входу у систему;
    f. avatar – фотокартка користувача;
    g. calories – денна норма калорій для користувача;
    h. protein – денна норма білків для користувача;
    i. triglyceride – денна норма жирів для користувача;
    j. carbohydrate – денна норма вуглеводів для користувача;
    k. gender – стать користувача. Використовуються при розрахунку денної норми калорій та БЖВ за формулою Харриса-Бенедікта;
    l. date_of_birth – дата народження користувача. Використовуються при розрахунку денної норми калорій та БЖВ;
    m. height – зріст користувача. Використовується при розрахунку денної норми калорій та БЖВ;

n. weight – вага користувача. Використовуються при розрахунку денної норми калорій та БЖВ.

2. Recipe – рецепт.
    a. id – унікальний ідентифікатор рецепту у форматі UUID, первинний ключ;
    b. title – назва рецепту;
    c. description – опис рецепту;
    d. image – фотокартка рецепту;
    e. author_id – унікальний ідентифікатор автора рецепту, зовнішній ключ;
    f. category_id – унікальний ідентифікатор категорії до якої належить рецепт, зовнішній ключ;
    g. cuisine_id – унікальний ідентифікатор країни походження рецепту, зовнішній ключ;
    h. menu_type_id – унікальний ідентифікатор типу меню до якого належить рецепт, зовнішній ключ;
    i. complexity – складність рецепту;
    j. cooking_time – час приготування рецепту;
    k. number_of_servings – очікувана кількість порцій рецепту;
    l. calories – калорійність у 100 гр. готової страви за рецептом;
    m. protein – вміст білків у 100 гр. готової страви за рецептом;
    n. triglyceride – вміст жирів у 100 гр. готової страви за рецептом;
    o. carbohydrate – вміст вуглеводів у 100 гр. готової страви за рецептом;
    p. rating – рейтинг рецепту.
3. Ingredient – інгредієнт.
    a. id – унікальний ідентифікатор інгредієнту у форматі UUID, первинний ключ;
    b. name – назва інгредієнту;
    c. calories - калорійність у 100 гр. інгредієнту;

d. protein – вміст білків у 100 гр. інгредієнту;
e. triglyceride – вміст жирів у 100 гр. інгредієнту;
f. carbohydrate – вміст вуглеводів у 100 гр. інгредієнту.

4. Recipe_item – допоміжна сутність, яка виступає у ролі складової списку інгредієнтів у рецепті. Дана сутність має збірний первинний ключ.
    a. recipe_id – унікальний ідентифікатор рецепту, первинний ключ;
    b. ingredient_id – унікальний ідентифікатор інгредієнту, первинний ключ;
    c. quantity – кількість інгредієнту;
    d. unit_id – розмірність кількості інгредієнту.
5. Unit – розмірність кількості інгредієнту.
    a. id – унікальний ідентифікатор розмірності кількості інгредієнту, первинний ключ;
    b. name – назва розмірності кількості інгредієнту.
6. Instruction_step – крок приготування рецепту.
    a. id – унікальний ідентифікатор кроку приготування рецепту, первинний ключ;
    b. recipe_id – унікальний ідентифікатор рецепту, зовнішній ключ;
    c. name – назва кроку приготування рецепту;
    d. description – опис кроку приготування рецепту;
    e. image – фотокартка кроку рецепту;
    f. serial_number – порядковий номер кроку у рецепті.
7. Tag – теги, які стосуються рецепту.
    a. id – унікальний ідентифікатор тегу, первинний ключ;
    b. name – назва тегу.
8. Recipe_tags – допоміжна сутність, яка пов’язує рецепти до тегів до яких вони відносяться. Дана сутність має збірний первинний ключ.
    a. recipe_id – унікальний ідентифікатор рецепту, первинний ключ;
    b. tag_id – унікальний ідентифікатор тегу, первинний ключ.

9. Category – категорія рецепту.
    a. id – унікальний ідентифікатор категорії рецепту, первинний ключ;
    b. name – назва категорії рецепту.
10. Cuisine – країна походження рецепту.
    a. id – унікальний ідентифікатор країни походження рецепту, первинний ключ;
    b. name – назва країни походження рецепту.
11. Menu_type – тип меню до якого відноситься рецепт.
    a. id – унікальний ідентифікатор типу меню до якого відноситься рецепт, первинний ключ;
    b. name – назва типу меню до якого відноситься рецепт.
12. Favorite_recipes – допоміжна сутність, яка відображає рецепти, які сподобались користувачу. Дана сутність має збірний первинний ключ.
    a. recipe_id – унікальний ідентифікатор рецепту, який сподобався, первинний ключ;
    b. user_id – унікальний ідентифікатор користувача, первинний ключ.
13. Unloved_recipes – допоміжна сутність, яка відображає рецепти, які не сподобались користувачу. Дана сутність має збірний первинний ключ.
    a. recipe_id – унікальний ідентифікатор рецепту, який не сподобався, первинний ключ;
    b. user_id – унікальний ідентифікатор користувача, первинний ключ.
14. Recipe_recommendation – допоміжна сутність, яка відображає рекомендації рецептів користувачу. Дана сутність має збірний первинний ключ.
    a. recipe_id – унікальний ідентифікатор рекомендованого рецепту, первинний ключ;

b. user_id – унікальний ідентифікатор користувача, первинний ключ;
c. date – дата рекомендації рецепту;
d. was_chosen – прапор, який показує, чи був рецепт обраний до приготування користувачем.

15. Comment – коментарі до рецепту.
    a. id – унікальний ідентифікатор коментаря до рецепту, первинний ключ;
    b. recipe_id – унікальний ідентифікатор рецепту, який прокоментували, зовнішній ключ;
    c. author_id – унікальний ідентифікатор автора коментаря, зовнішній ключ;
    d. text – текст коментаря;
    e. image – фотокартка закріплена за коментарем;
    f. date – дата та час створення коментаря;
    g. likes – кількість лайків до коментаря;
    h. dislikes – кількість дислайків до коментаря.

Зв’язки між переліченими сутностями представлені у схемі відносин на рис 3.15.

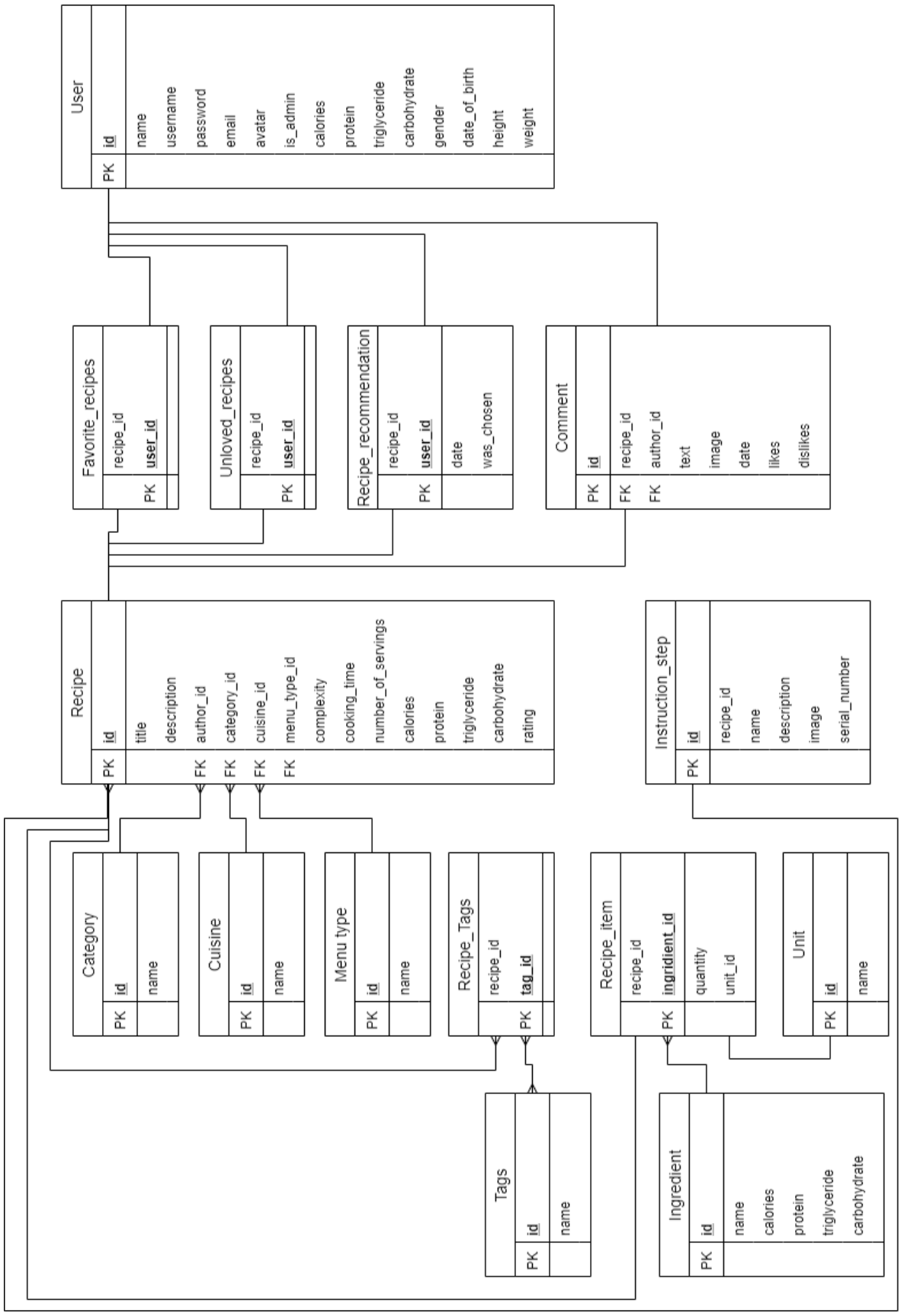


Рис. 3.10. Схема структури бази даних веб-додатку

3.6. Алгоритм отримання даних рецептів із веб-сторінок

Однією із задач, з якими має справу розроблюваний веб-додаток є пошук нових рецептів, їх обробка та збереження в БД для формування рекомендацій страв своїм користувачам. Алгоритм пошуку та оброблення інформації про рецепти було розділено на два окремих:

- пошук інформації у відкритих джерелах мережі Інтернет;
- оброблення і збереження знайдених рецептів в БД.

Розглянемо перший алгоритм. Його було реалізовано в рамках сервісу збору даних рецептів. Завдання цього сервісу – перебрати всі доступні сторінки обраних веб-сайтів в мережі Інтернет та знайти ті, на яких знаходяться рецепти.

Алгоритм можна розбити на кроки(рис. 4.1):

1. Системою ініціюється початок роботи алгоритму. За розкладом в порожню чергу завдань AWS SQS на пошук рецептів додається завдання на обробку стартової сторінки веб-ресурсу.
2. Дії 2-7 виконуються кожним екземпляром програми, доки черга не буде порожньою.
3. Екземпляр сервісу збору даних рецептів дістає із черги завдань AWS SQS наступний URL та завантажує сторінку за цим адресом.
4. Якщо в сервісі існує механізм оброблення даного виду сторінок, прив’язаний до URL сторінки, то сервіс виконує дії, описані в пункті 5, якщо ні – переходить до пункту 6.
5. Знайти на сторінці відомості про рецепт та покласти необроблені текстові дані в чергу завдань обробників рецептів AWS SQS.
6. Знайти на сторінці всі нові посилання та покласти їх в чергу на оброблення.
7. Перейти до пункту 3;
8. Завершити роботу.

Рис. 4.1. Схема алгоритму роботи сервісу збирання даних рецептів

3.7. Алгоритм оброблення даних рецептів

Іншим важливим алгоритмом, який дозволяє системі автоматично наповнювати БД рецептів, є алгоритм обробки текстових даних, отриманих зборі даних із мережі Інтернет. Алгоритм оброблення даних рецептів полягає у відокремленні даних зі сторінок-джерел та обчисленню калорійності і складових, використовуючи збережені в БД відомостей про кожен інгредієнт, наявний у рецепті (рис. 4.2).

Рис. 4.2. Алгоритм обробки даних рецептів

На основі текстових даних, які містять HTML розмітку сторінки, сервіс обробки рецептів має дістати наступні дані:

1. Інгредієнти та їх кількість.
2. Фотографії готової страви.
3. Оцінка користувачами веб-ресурсу джерела даного рецепту.
4. Покрокову інструкцію приготування.

Для формування правил отримання цієї інформації зі сторінок веб-сайтів, було проаналізовано принципи їх побудови всіма використаними джерелами рецептів. На основі цих даних було сформовано правила, як діставати потрібні дані із кожного веб-сайту.

Оскільки HTML сторінки представляють собою XML дерево, було застосовано опис правил доступу до даних на основі мови запитів до XML-документа XPath [30]. XPath дозволяє, описуючи шляхи доступу до окремих вузлів DOM-дерева [31] отримувати дані, які там зберігаються.

Наступна трансформація даних полягала у відокремлені текстових даних від усіх HTML тегів. Після виконання цієї дії, сервіс може оперувати строками, які містить лише текстові дані рецепту. Із опису інгредієнтів за допомогою регулярних виразів [32] отримується кількість та назва кожного з них. Володіючи інформацією про кількість потрібного інгредієнту та його назву, сервіс формує запит до БД, який дістає дані про приблизний вміст БЖВ та калорій у готовій страві. На основі цих чисел формується відомість про поживні складові рецепту та його характеристики. Отримані дані заносяться в БД додатку. На цьому виконання алгоритму завершено.

4. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНИХ ЗАСОБІВ

4.1. Опис процесу автоматизації розгортки веб-додатку

Важливим кроком, який має бути виконаний при розгортці нової версії додатку є процес доставки ПЗ в оточення, де воно буде виконуватися. Оскільки оточенням, в якому буде працювати веб додаток для підтримки формування раціону користувача, буде платформа хмарних обчислень AWS, процес розгортки ПЗ має включати оновлення залежностей кожного сервісу та зборка контейнерів із кодом сервісу.

Авторами було використано підхід до розроблення архітектури ПЗ IaC (укр. «інфраструктура як код») [33], який передбачає розгортання та керування мережевими і обчислювальними ресурсами платформ шляхом їх опису у вигляді програмного коду. Для формування відповідного коду використано розробку компанії HashiCorp для реалізації IaC – Terraform [34]. Отримані інструкції інтегровано до системи безперервної доставки коду Gitlab CI [35], за допомогою чого досягнуто автоматичного розгортання веб-додатку після кожного виписку нової версії ПЗ.

4.2. Опис інтерфейсу користувача

Інтерфейс даного веб-застосунку виконано у мінімалістичному стилі, є інтуїтивно-зрозумілим та легким у використанні.

Дизайн сайту був виконано у трьох основних кольорах: #BC2F3C (відтінок жовтого), #777777 (відтінок сірого) та #FFFFFF (фоновий білий). Всі інші кольори, які розміщені на сайті, отримані внаслідок зміни прозорості основних кольорів. Використання не більше ніж 3-4 кольорів, є гарною практикою при виконанні дизайну. Жовтий колір був обраний у якості основного кольору зважаючи на специфіку веб-застосунку. Даний теплий колір стимулює хороший апетит та позитивні емоції.

Шапка (хедер) сайту містить логотип, меню, глобальний пошук по сайту та секцію кнопок переходу до особистого кабінету (у разі якщо

користувач авторизований), або кнопки авторизації. Логотип містить назву сайту. Меню складається з наступних кнопок: «Рецепти» – перехід на сторінку з каталогом рецептів, «Що мені хочеться?» – перехід на сторінку з рекомендаціями. Також вигляд шапка сайту буде відрізнятися в залежності авторизовано користувач чи ні. Якщо користувач неавторизований у системі, то у шапці буде розміщуватися кнопка «Увійти». Якщо ж користувач авторизований в системі, то у шапці будуть відображатися кнопку «Збережені рецепти» – перехід на сторінку зі збереженими рецептами, та кнопку переходу до особистого кабінету. Приклад представленого хедера наведено на рис. 4.3.

Рис. 4.3. Шапка сайту авторизованого користувача

Для реєстрації та авторизації були використані спливаючі вікна. Форма авторизації містить наступні поля: логін/електронна адреса та пароль, а реєстрації – логін, електрона пошта, пароль та підтвердження паролю. Також кожна форма містить кнопку показати/скрити ведений пароль, яка розташована поруч з відповідним полем. Між представленими вікнами можна швидко переключатися за допомогою кнопок «Увійти» та «Зареєструватися». Приклад представлених спливаючих вікон наведено на рис. 4.4.

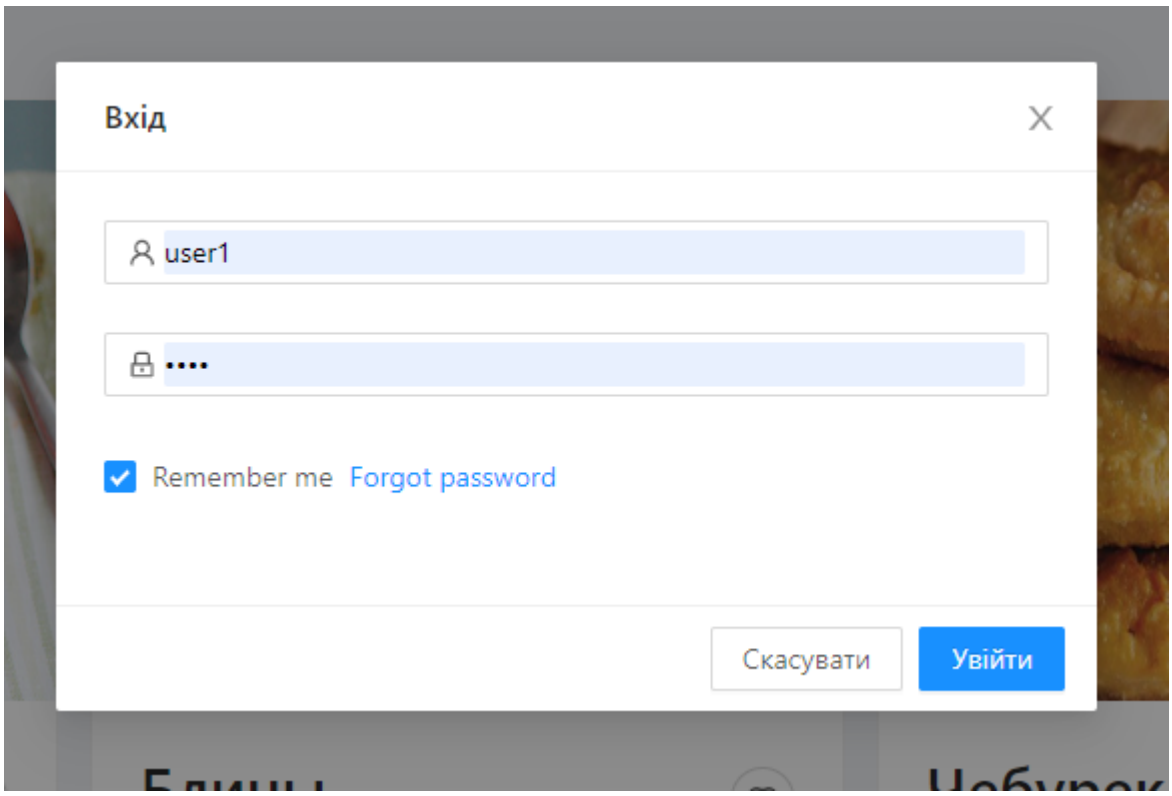


Рис. 4.4. Вікно авторизації

Сторінка «Рецепти» відповідає за перегляд каталогу рецептів, які представленні на даному сайті. Дана сторінка поділена на дві секції: з ліва - меню для фільтрації, з права – список рецептів. Меню для фільтрації, поділено на декілька підпунктів, кожен з яких відповідає за окремий параметр фільтрації. Кожен рецепт з списку рецептів представлено карткою, яка містить основну інформацію: назву рецепту, зображення готової страви, теги які стосуються даного рецепту, список інгредієнтів, кількість порцій, приблизній час приготування, кнопку «Зберегти», кількість лайків та дизлайків. Також секція з рецептами містить форму пошуку за назвою рецепту, та кнопки переходу на інші сторінки з рецептами. Приклад сторінки Рецепти наведено на рис. 4.5.

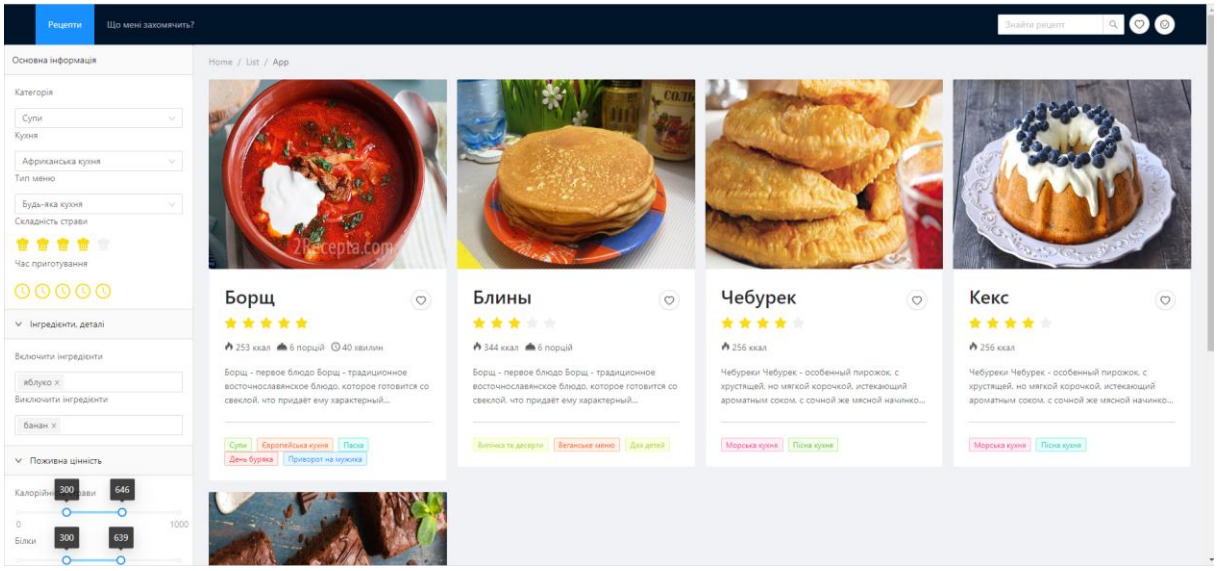


Рис. 4.5. Сторінка перегляду каталогу рецептів

Сторінка з конкретним рецептом містить наступну структуру: короткий опис рецепту (назва рецепту, зображення готової страви, теги які стосуються даного рецепту, кількість порцій, приблизній час приготування, кнопку «Зберегти», кількість лайків та дизлайків, опис енергетичної цінності на одну порцію(кількість калорій та БЖВ), список інгредієнтів, інструкція приготування, секція коментарів. Приклад сторінки з конкретним рецептом наведено на рис. 4.6, рис. 4.7 , рис. 4.8 та рис. 4.9

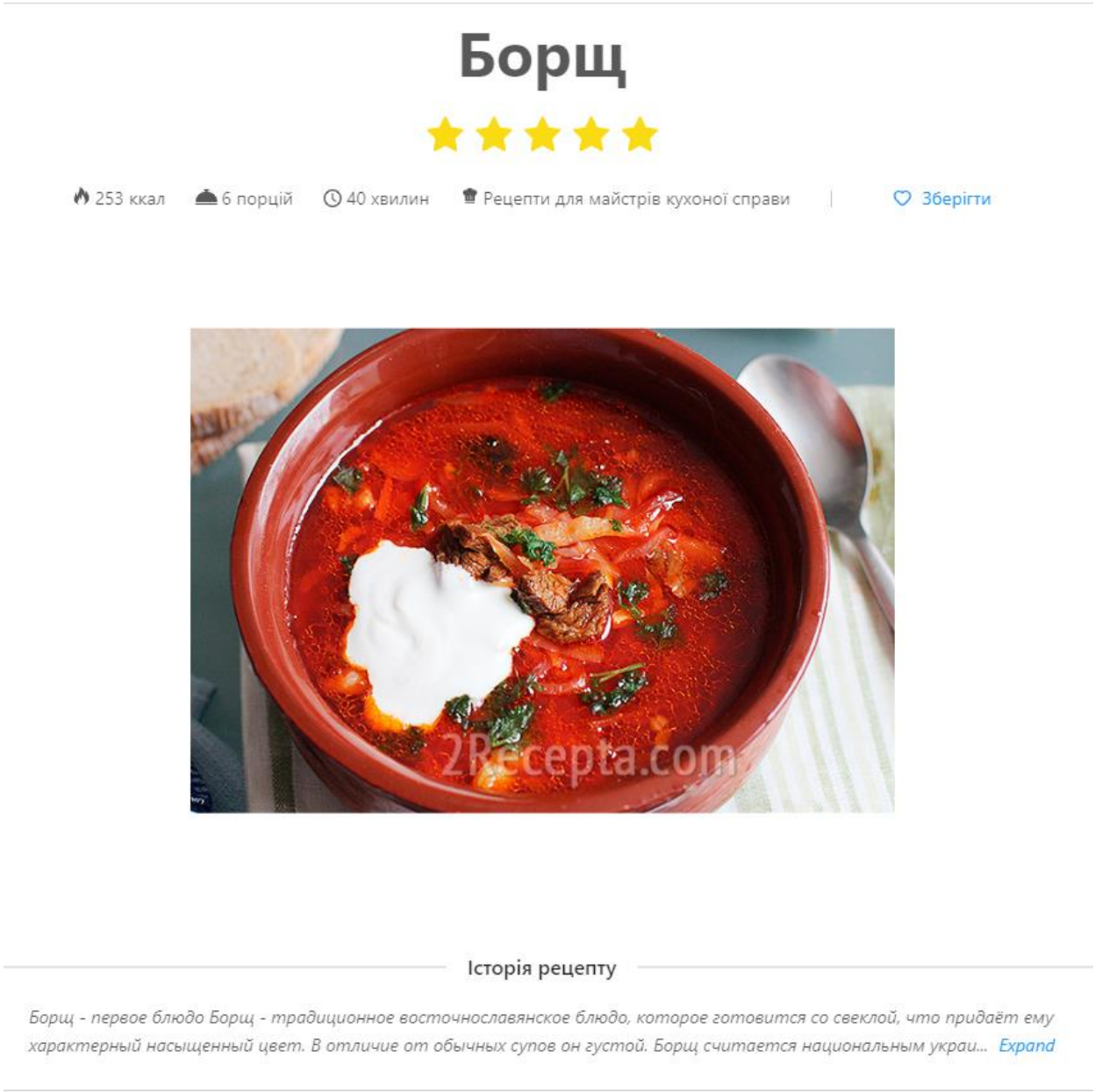


Рис. 4.6. Сторінка перегляду конкретного рецепту. Секція основної інформації

Енергетична цінність на порцію

Калорійність	Білки	Жири	Вуглеводи
253 ккал	33 грам	23 грам	45 грам

** Калорійність розрахована для сирих продуктів*

Інгредієнти на 6 порцій

Інгредієнти	Кількість
Шпинат	150 гр.
Молоко	1 шт.
Куриные яйца	2 шт.
Соль	1 чайн. л.

Рис. 4.7. Сторінка перегляду конкретного рецепту. Секції «Енергетична цінність» та «Інгредієнти»

Інструкція приготування

40 хвилин Друкувати

Крок 1:

Подготовим продукты.

Крок 2:

Если у вас замороженный шпинат- ему нужно дать время оттаять. Если свежий (как у меня) - его нужно хорошо промыть под проточной водой, оборвать лишние стебельки и бланшировать в кпящей воде 15 секунд, после хорошо отжать.

Крок 3:

Теперь измельчение. Если хотите - можете воспользоваться погружным блендером для измельчения шпината. Если есть блендер стационарный - тоже очень хорошо. Нам нужно хорошо измельчить листья. В чашу блендера помещаем шпинат и яйца и хорошо измельчаем.

Крок 4:

Проверяйте консистенцию теста - оно должно быть жидким как на тонкие блины.

Рис. 4.8. Сторінка перегляду конкретного рецепту. Секція «Інструкція приготування»

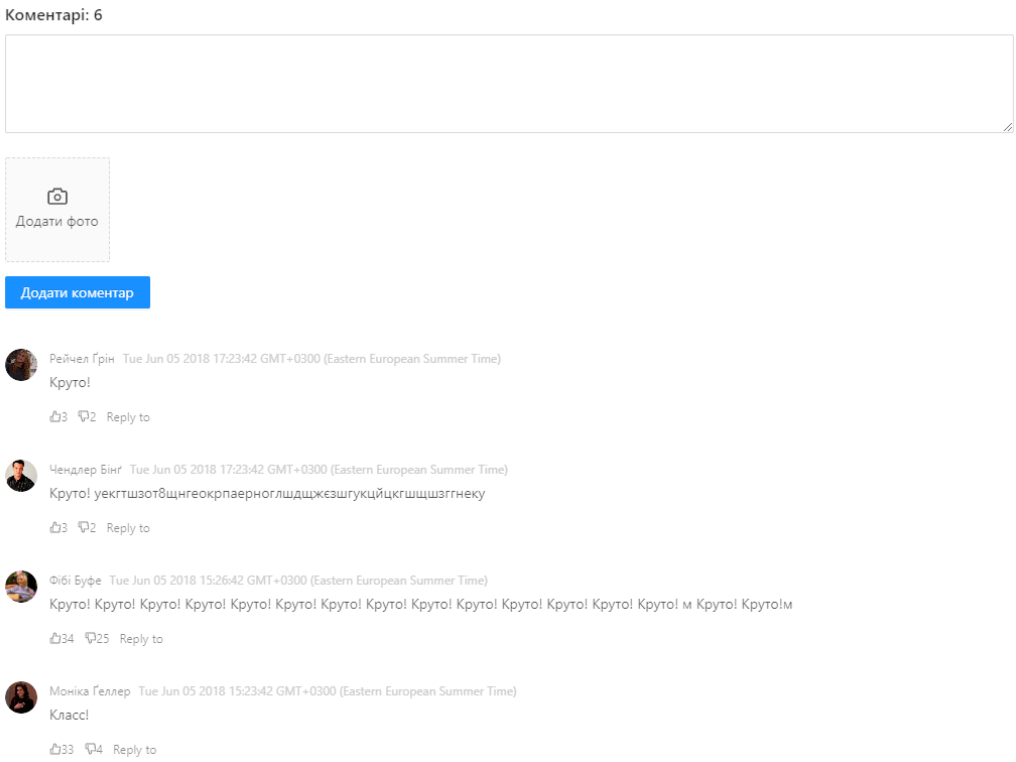


Рис. 4.9. Сторінка перегляду конкретного рецепту. Секції «Коментарі»

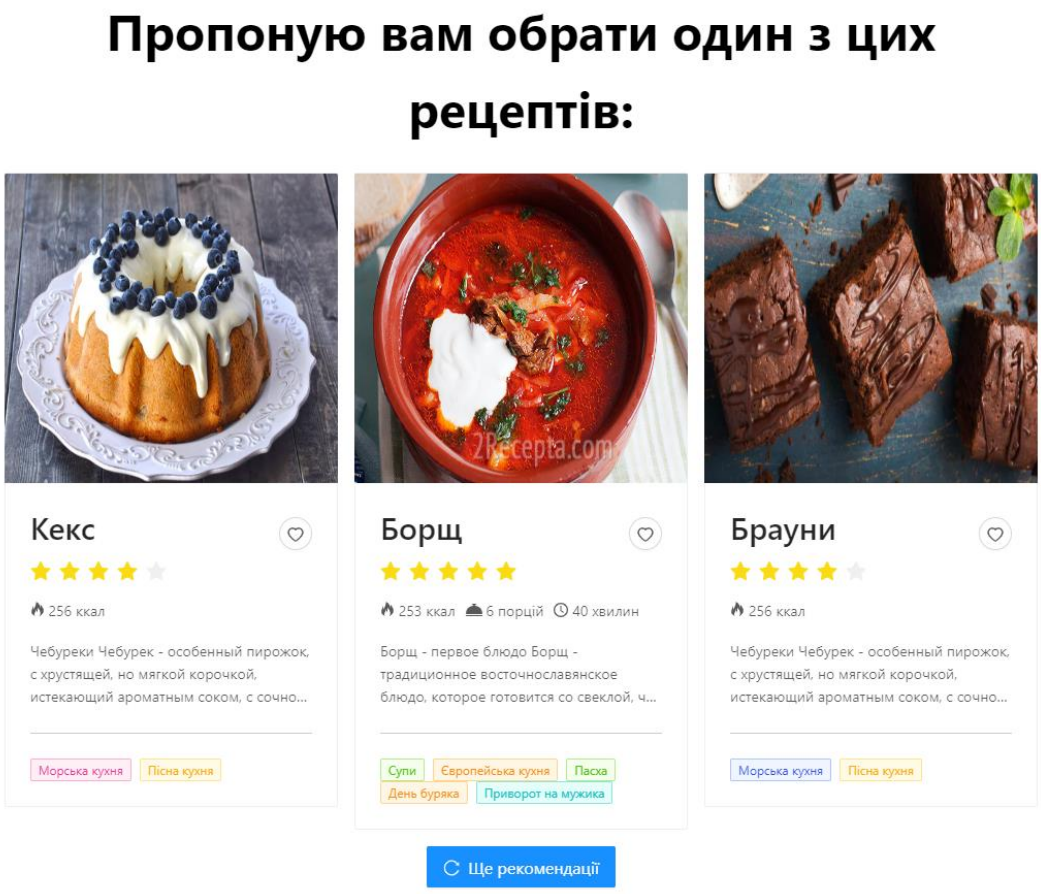


Рис. 4.10. Сторінка з рекомендаціями

Сторінка з рекомендаціями містить список з 3 рецептів. Кожен рецепт представлено у вигляді картки. Після блоку рецептів наявна кнопка «Ще рекомендації», яка ініціює підбор нових рекомендацій. Приклад сторінки з рекомендаціями представлено на рис. 4.10.

Сторінка зі збереженими рецептами виглядає як і сторінка Рецепти. Єдина різниця тільки в тому, що на цій сторінці відображаються тільки рецепти, які користувач відмітив як збережені.

Перейдемо до сторінки особистого кабінету користувача. «Профіль» – сторінка з особистими даними користувача, на ній містяться наступні секції: особиста інформація користувача (фото профіля користувача, повне ім’я, логін, електронна адреса, стать, дата народження, зріст та вага), інформація про рекомендовану кількість калорій та БЖВ та форма зміни паролю. Приклад роботи сторінки профілю користувача знаходиться на рис. 4.11.

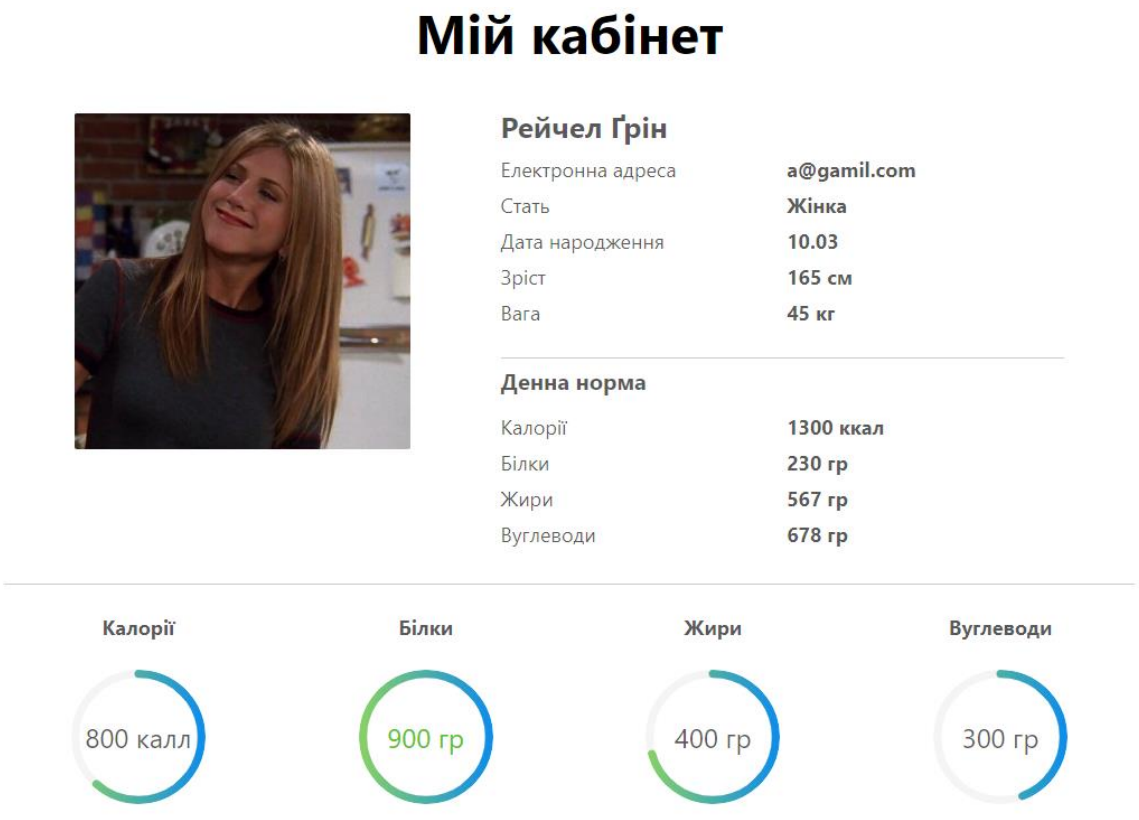


Рис. 4.11. Сторінка профілю користувача

Основною метою при розробленні інтерфейсу веб-додатку для підтримки формування раціону було забезпечити простоту використання та інтуїтивну зрозумілість. Клієнтська частина не перевантажена анімацією та яскравими кольорами.

4.3. Опис процесу тестування

На базі вимог до розроблюваного додатку, які були представлені у підрозділі 3, було сформовано набір тест-кейсів (табл. 4.1). Для перевірки правильності роботи застосунку було обрано димове (smoke) тестування.

Таблиця 4.1

Тестові випадки (тест-кейси)

№ п/п	Назва тест-кейсу	Опис	Очікуваний результат
1	Авторизація користувача в системі	1. Натиснути кнопку авторизації. 2. Ввести логін/електрону пошту та пароль у форму входу. 3. Натиснути кнопку «Увійти».	1. Система подає спливаюче вікно авторизації. 2. Пароль ховається за крапками. 3. Користувач знаходиться на тій же сторінці, в шапці є кнопка переходу до особистого кабінету користувача.
2	Пошук рецепту	1. Перейти на сторінку «Рецепти». 2. Налаштувати фільтрацію.	1. Система формує список рецептів за заданими параметрами.

Продовження табл. 4.1

		3. Натиснути на кнопку «Знайти рецепти». 4. Обрати рецепт із списку запропонованих.	2. Система відображає знайдені рецепти. 3. Після обрання рецепту, система переходить на конкретний рецепт. Та відображає інформацію про нього.
3	Перегляд рецепту	1. Перейти на сторінку із обраним рецептом.	1. Система коректно відображає дані про рецепт та його складові. 2. Система розраховує рейтинг рецепту. 3. Система відображає відгуки користувачів про рецепт.
4	Додати рецепт до списку улюблених	1. Перейти на сторінку із обраним рецептом. 2. Натиснути на кнопку «Додати до списку обраних».	1. Коректне представлення сторінки. 2. Рецепт потрапляє до списку обраних рецептів даного користувача.
5	Додати рецепт до чорного списку	1. Перейти на сторінку із обраним рецептом. 2. Натиснути на кнопку «Додати до чорного списку».	1. Коректне представлення сторінки. 2. Рецепт потрапляє до чорного списку рецептів даного користувача. 3. Рецепт більше не відображається в

Продовження табл. 4.1

			щоденних рекомендаціях.
6	Залишити відгук на рецепт	1. Перейти на сторінку перегляду рецепту 2. Натиснути кнопку оцінити. 3. Поставити оцінку рецепту. 4. Залишити текстовий коментар до рецепту 5. Додати фотографії до відгуку	1. Правильне представлення даних про рецепт. 2. Коректне представлення форм для залишення коментаря до рецепту. 3. Коментар користувача відображається на сторінці перегляду рецепту.

У табл. 4.1 були наведені лише основні тест-кейси, які в повній мірі покривають роботу всього функціоналу веб-додатку для підтримки формування раціону. Після тестування програмного забезпечення за допомогою даних тест кейсів відхилення від висунутих вимог не було виявлено.

4.4. Подальші покращення

Наступним кроком для вдосконалення та розширення даної системи є додання нейронної мережі для побудови рекомендацій рецептів для користувачів та побудова моделі для більш точної обробки текстової інформації зі сторінок сайтів, які вміщують рецепти.

Також цікавим розширенням може бути додання онлайн-чатів, що забезпечить спілкування між користувачами, які можуть давати поради один одному щодо особливостей приготування страв за обраними рецептами.

Ще одним варіантом для забезпечення монетизації даного веб-застосунку є додання персоналізації до дизайну сторінок веб сайту, побудова додаткової секції на сторінці рецепту, де буде розміщуватися інформація про необхідне обладнання для приготування рецепту із посиланнями на веб-сайти партнерів проєкту, які продають це кухонне знаряддя.

ВИСНОВКИ

Метою даного дипломного проєкту є розроблення веб-додатку для підтримки формування раціону користувача на основі платформи AWS.

Для реалізації проєкту було знайдено існуючі програмні рішення, які вирішують схожу проблему. Їх аналіз показав, що аналогу даному веб-застосунку немає. Тому створення системи даного типу є доцільним. Було сформовано основні вимоги до технологій та засобів розроблення ПЗ, Серед доступних на сьогодні інструментів, було обрано найбільш відповідні для даного проєкту та обґрунтовано їх вибір, а саме: C#, .Net, AWS Lambda, JavaScript та PostgreSQL.

В третьому розділі даної пояснювальної записки було описано функціональні та не функціональні вимоги до розроблюваного веб-додатку. Їх дотримання є обв’язкою умовою для того аби вважати , що програмне рішення було реалізоване в повному обсязі. Після чого було розроблено структуру веб-додатку та бази даних.

В четвертому розділі розглянуто основні алгоритми оброблення інформації, сутності бази даних та компоненти інтерфейсу які були реалізовані в процесі роботи над даним веб-додатком. Також було наведено тестові випадки, які перевірялися за допомогою димового тестування для підтвердження відповідності веб-додатку функціональним вимогам. В результаті чого відхилень не було. Наведено рекомендації щодо подальшого розширення розроблюваного веб-застосунку.

Результатом роботи є працюючий веб-застосунок, який дозволяє формувати свій раціон користувачам за допомогою гнучкого пошуку серед відомих системі рецептів страв та персональних рекомендацій.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

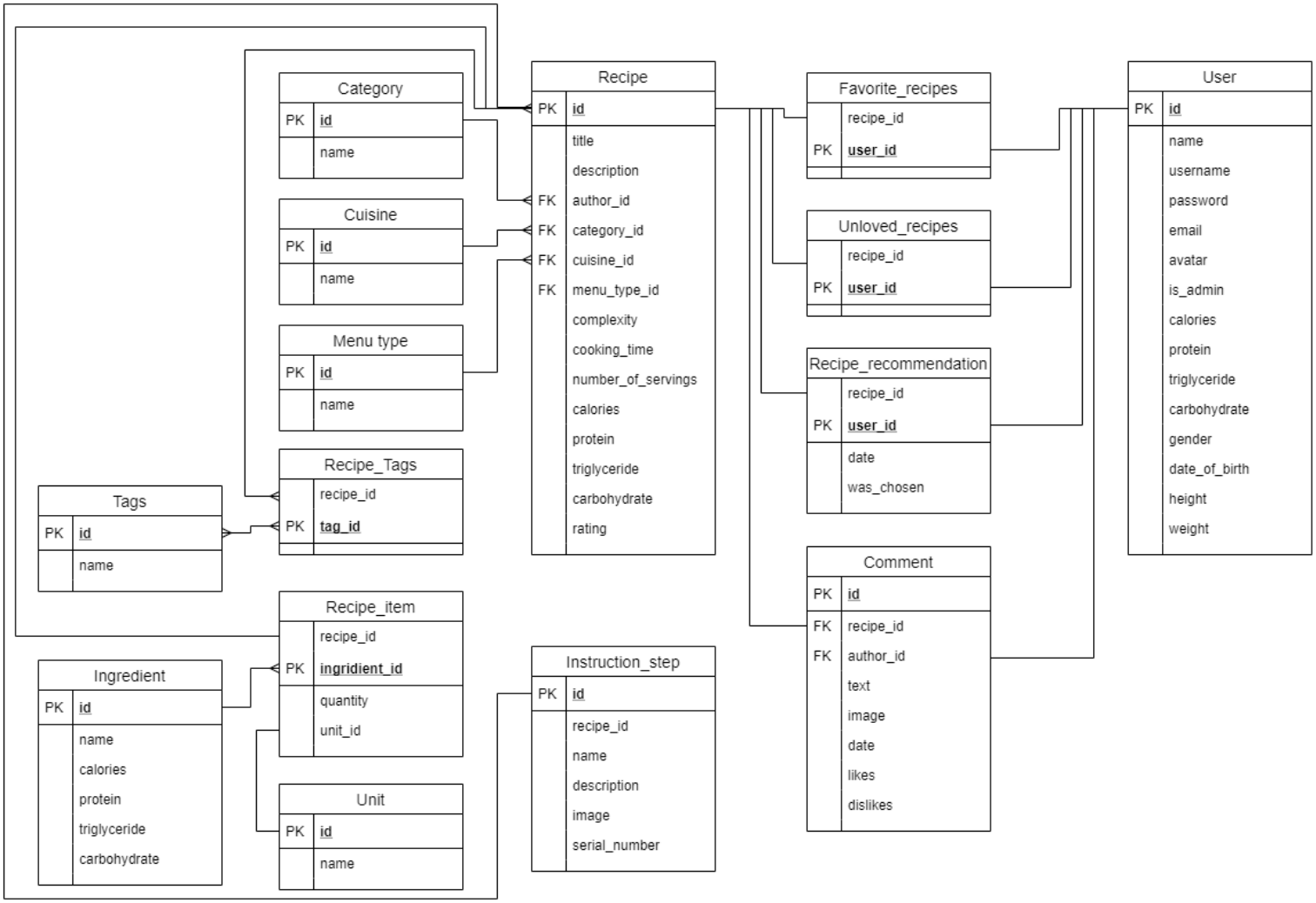
1. Диетолог [Електронний ресурс]. — Режим доступу: https://ru.wikipedia.org/wiki/Диетолог/
2. Cara: Mood, Poop & Food Tracker [Електронний ресурс]. — Режим доступу: https://foodandnutrition.org/may-june-2017/cara-mood-poop-food-tracker-version-1-20/
3. Rise Up + Recover app [Електронний ресурс]. — Режим доступу: https://www.healthnavigator.org.nz/apps/r/rise-up-plus-recover-app/
4. Еда [Електронний ресурс]. — Режим доступу: https://eda.ru/
5. What is Python: [Електронний ресурс]. – Режим доступу до ресурсу: https://www.pythonforbeginners.com/learn-python/what-is-python/
6. Best Programming Language to Learn in 2020 (for Job & Future): [Електронний ресурс]. — Режим доступу: https://hackr.io/blog/best-programming-languages-to-learn-2020-jobs-future
7. Advantages and Disadvantages of Python Programming Language [Електронний ресурс]. — Режим доступу: https://medium.com/@mindfiresolutions.usa/advantages-anddisadvantages-of-python-programming-language-fd0b394f2121/
8. The Python Package Index (PyPI) [Електронний ресурс]. — Режим доступу: https://docs.python.org/3/distutils/packageindex.html
9. C Sharp (programming language): [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/
10. ASP.NET documentation [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1/
11. PHP [Електронний ресурс]. — Режим доступу: https://www.php.net/
12. WHAT IS THE BEST LANGUAGE FOR WEB DEVELOPMENT IN 2020? [Електронний ресурс]. — Режим доступу: https://mikkegoes.com/best-language-for-web-development/

13. What is JavaScript [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/enUS/docs/Learn/JavaScript/First_steps/What_is_JavaScript
14. Angular [Електронний ресурс]. — Режим доступу: https://angular.io/
15. TypeScript [Електронний ресурс]. — Режим доступу: https://www.typescriptlang.org/
16. React [Електронний ресурс]. — Режим доступу: https://reactjs.org/
17. Vue.js [Електронний ресурс]. — Режим доступу: https://vuejs.org/
18. Строгая типизация для приложений Vue.js на TypeScript [Електронний ресурс] — Режим доступу: https://habr.com/ru/post/351216/
19. MySQL [Електронний ресурс] — Режим доступу:https://www.mysql.com/
20. PostgreSQL [Електронний ресурс] — Режим доступу: https://www.postgresql.org/
21. Appendix D. SQL Conformance [Електронний ресурс] — Режим доступу: https://www.postgresql.org/docs/current/features.html
22. CLIENT – SERVER ARCHITECTURE [Електронний ресурс] — Режим доступу: https://anirudhrajmohan.wordpress.com/2018/08/21/client-server-architecture/
23. What is AJAX? [Електронний ресурс] — Режим доступу: https://medium.com/@jameshamann/what-is-ajax-a63ca1a68d99. – (10.3.2020).
24. What is a CDN? | How do CDNs work? [Електронний ресурс] — Режим доступу: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/
25. FaaS (Function-as-a-Service) [Електронний ресурс] — Режим доступу: https://www.ibm.com/cloud/learn/faas
26. What is REST — A Simple Explanation for Beginners, Part 1: Introduction [Електронний ресурс] — Режим доступу: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f

27. Why do we need the JSON Web Token (JWT) in the modern web? [Електронний ресурс]. — Режим доступу: https://medium.com/swlh/why-do-we-need-the-json-web-token-jwt-in-the-modern-web-8490a7284482
28. Amazon API Gateway [Електронний ресурс]. — Режим доступу: https://aws.amazon.com/ru/api-gateway/
29. What is a RESTful API? [Електронний ресурс]. — Режим доступу: https://medium.com/@lazlojuly/what-is-a-restful-api-fabb8dc2afeb
30. XPath Tutorial [Електронний ресурс]. — Режим доступу: https://www.w3schools.com/xml/xpath_intro.asp
31. DOM tree [Електронний ресурс]. — Режим доступу: https://javascript.info/dom-nodes
32. Регулярные выражения. [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Regular_Expressions
33. Infrastructure as Code (IaC) – What is it? [Електронний ресурс]. — Режим доступу: https://medium.com/swlh/infrastructure-as-code-iac-what-is-it-c173457393c7
34. Introduction to Terraform — Режим доступу до ресурсу: https://www.terraform.io/intro/index.html
35. GitLab CI/CD [Електронний ресурс]. — Режим доступу: https://docs.gitlab.com/ee/ci/

ДОДАТКИ

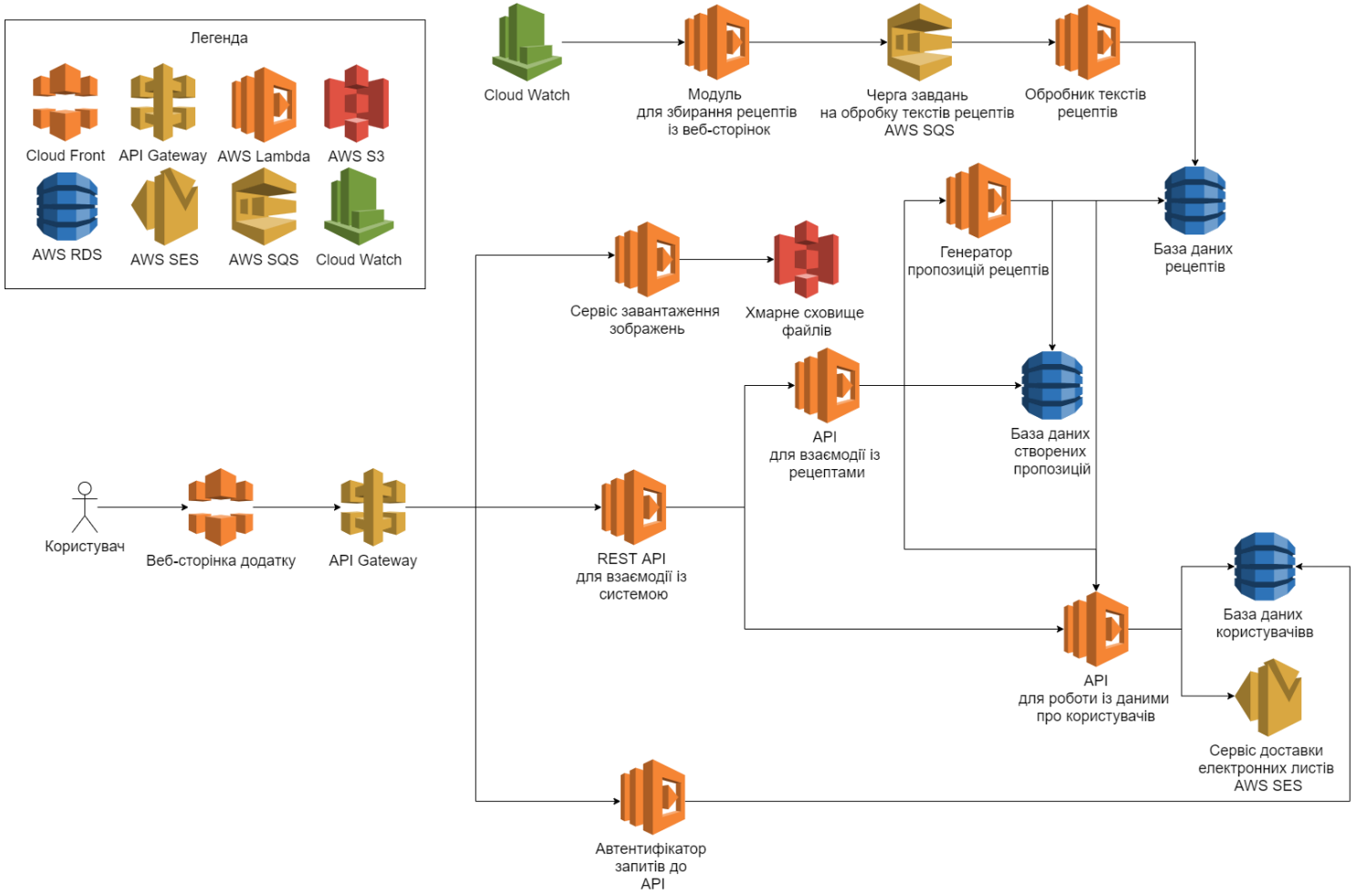
Додаток 1
Копії графічних матеріалів



ДП.045440-06-99. Веб-додаток для підтримки формування раціону користувача на основі платформи AWS. Структура бази даних. ERD-діаграма

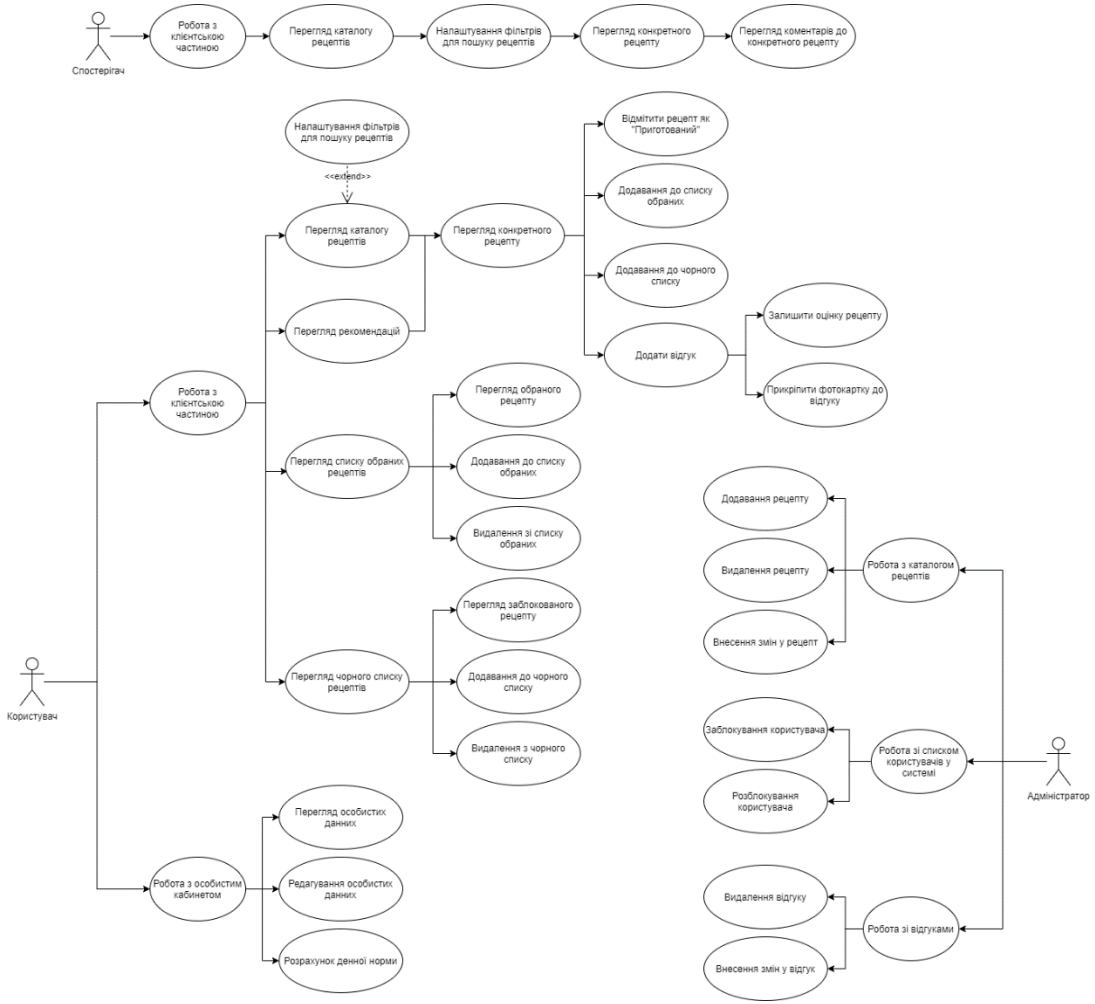
ДП.045440-07-99. Веб-додаток для підтримки формування раціону користувача на основі платформи AWS. Алгоритм отримання даних рецептів із веб-сторінок. Схема алгоритму

СХЕМА СТРУКТУРИ СЕРВРНОЇ ЧАСТИНИ ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS



Карпенко О.О., група КП-61

ДІАГРАМА ПРЕЦЕДЕНТІВ ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS



Карпенко О.О., група КП-61

Додаток 2
Лістинг програми

```
import React from 'react';
import { RouteComponentProps } from '@reach/router';
import { Layout, Row, Col, Avatar, Divider, Progress } from 'antd';
import { useStoreState } from '../state';

const { Content } = Layout;

export const Profile: React.FC<RouteComponentProps> = () => {
  const user = useStoreState((state) => state.user.user);
  console.log(user);

  function getPercent(value: number | undefined, maxValue: number | undefin
ed): number | undefined {
    if (value && maxValue) {
      return (value / maxValue) * 100;
    }
    return undefined;
  }

  return (
    <>
      <Layout style={{ padding: '0 24px 24px', backgroundColor: '#FFEBC2' }
}>
        <Content
          className='site-layout-background'
          style={{
            marginLeft: '22%',
            marginRight: '22%',
            padding: '0 1% 1%',
            backgroundColor: 'white',
            alignContent: 'center',
          }}
        >
          <div style={{ textAlign: 'center', marginBottom: '25px' }}>
            <b style={{ fontSize: 48, color: 'black' }}>Мій кабінет</b>
          </div>
          <Row gutter={[4, 4]} justify="space-around">
            <Col span={6}>
              <Avatar shape='square' size={300} src={user?.avatar} style={{
 marginTop: '10px' }} />
            </Col>
            <Col span={12}>
              <Row gutter={[8, 4]} style={{ fontSize: 26, marginTop: '0px',
 paddingTop: '0px' }}>
                <Col span={12}>
                  <b>{user?.name}</b>
                </Col>
              </Row>
              <div style={{ fontSize: 18 }}>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Електронна адреса</Col>
                  <Col span={12}>
                    <b>{user?.email}</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Стать</Col>
                  <Col span={12}>
                    <b>{user?.gender}</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Дата народження</Col>
                  <Col span={12}>
```

```
                    <b>{user?.date_of_birth}</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Зріст</Col>
                  <Col span={12}>
                    <b>{user?.height} см</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Вага</Col>
                  <Col span={12}>
                    <b>{user?.weight} кг</b>
                  </Col>
                </Row>
                <Divider style={{ marginBottom: '4px' }} />
                <Row
                  gutter={[8, 4]}
                  style={{ fontSize: 20, marginBottom: '10px', paddingTop:
'0px' }}
                >
                  <Col span={12}>
                    <b>Денна норма</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Калорії</Col>
                  <Col span={12}>
                    <b>{user?.calories} ккал</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Білки</Col>
                  <Col span={12}>
                    <b>{user?.protein} гр</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Жири</Col>
                  <Col span={12}>
                    <b>{user?.carbohydrate} гр</b>
                  </Col>
                </Row>
                <Row gutter={[8, 4]}>
                  <Col span={12}>Вуглеводи</Col>
                  <Col span={12}>
                    <b>{user?.triglyceride} гр</b>
                  </Col>
                </Row>
              </div>
            </Col>
          </Row>
          <Divider />

          <Row gutter={[16, 24]} justify="space-around">
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center', fontSize: 18 }}>
                <b>Калорії</b>
              </div>
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center', fontSize: 18 }}>
                <b>Білки</b>
              </div>
```

```
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center', fontSize: 18 }}>
                <b>Жири</b>
              </div>
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center', fontSize: 18 }}>
                <b>Въглеводи</b>
              </div>
            </Col>

            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center' }}>
                <Progress
                  type='circle'
                  strokeColor={{
                    '0%': '#108ee9',
                    '100%': '#87d068',
                  }}
                  percent={getPercent(user?.todayCalories, user?.calories)}
                  format={() => `${user?.todayCalories} калл`}
                />
              </div>
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center' }}>
                <Progress
                  type='circle'
                  strokeColor={{
                    '0%': '#108ee9',
                    '100%': '#87d068',
                  }}
                  percent={getPercent(user?.todayProtein, user?.protein)}
                  format={() => `${user?.todayProtein} гр`}
                />
              </div>
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center' }}>
                <Progress
                  type='circle'
                  strokeColor={{
                    '0%': '#108ee9',
                    '100%': '#87d068',
                  }}
                  percent={getPercent(user?.todayCarbohydrate, user?.carboh
ydrate)}
                  format={() => `${user?.todayCarbohydrate} гр`}
                />
              </div>
            </Col>
            <Col className='gutter-row' span={6}>
              <div style={{ textAlign: 'center' }}>
                <Progress
                  type='circle'
                  strokeColor={{
                    '0%': '#108ee9',
                    '100%': '#87d068',
                  }}
                  percent={getPercent(user?.todayTriglyceride, user?.trigly
ceride)}
                  format={() => `${user?.todayTriglyceride} гр`}
                />
```

```
            </div>
          </Col>
        </Row>
      </Content>
    </Layout>
  </>
 );
};
import React, { useState, useCallback } from 'react';
import { RouteComponentProps, useParams } from '@reach/router';
import { Layout, Button, Form, Modal, InputNumber, Row, Col } from 'antd';
import {
 EnergyValueSection,
 IngredientsSection,
 InstructionSection,
 MainInfoSection,
 TagsSection,
} from '../components/recipe';
import { CommentSection } from '../components/comment';
import { RECIPESLIST } from '../mocks';
import { useStoreState } from '../state';
import { useCommentsFetch } from '../effects/useCommentsFetch';
import { PlusCircleOutlined } from '@ant-design/icons';

const { Content } = Layout;

export const Recipe: React.FC<RouteComponentProps> = () => {
 const user = useStoreState((state) => state.user.user);
 let params = useParams();
 let recipeId: string = params.id;
 let recipe = RECIPESLIST.find((x) => x.id === recipeId);
 const [isOpen, setIsOpen] = useState(false);
 const [form] = Form.useForm();

 const comments = useStoreState((state) => state.comments.comments[recipeI
d]);
 useCommentsFetch({ id: recipeId });

 const onSubmit = useCallback((values) => {
   user?.addDish(values.quantity, recipe!.calories, recipe!.protein, recip
e!.triglyceride, recipe!.carbohydrate);
   closePopup();
 }, []);

 const closePopup = useCallback(() => {
   form.resetFields();
   setIsOpen(false);
 }, [form]);

 return (
   <>
     {recipe && (
       <Layout style={{ padding: '0 24px 24px', backgroundColor: '#FFEBC2'
 }}>
         <Content
           className='site-layout-background'
           style={{
             marginLeft: '22%',
             marginRight: '22%',
             padding: '0 1% 1%',
             backgroundColor: 'white',
           }}
         >
           <MainInfoSection
```

```
              title={recipe.title}
              rating={recipe.rating}
              calories={recipe.calories}
              numberOfServings={recipe.numberOfServings}
              cookingTime={recipe.cookingTime}
              complexity={recipe.complexity}
              images={recipe.images}
              description={recipe.description}
            />
            <EnergyValueSection
              caloricity={recipe.calories}
              protein={recipe.protein}
              triglyceride={recipe.triglyceride}
              carbohydrate={recipe.carbohydrate}
            />
            <IngredientsSection ingredients={recipe.ingredients} portion={6
} />
            <InstructionSection instruction={recipe.instruction} cookingTim
e={40} />
            <div style={{ textAlign: 'center' }}>
              <Button
                shape='round'
                icon={<PlusCircleOutlined />}
                size='large'
                style={{ color: '#E09600', borderColor: '#E09600' }}
                onClick={() => setIsOpen(true)}
              >
                Врахувати у денну норму
              </Button>
              <Modal
                visible={isOpen}
                title='Врахувати у денну норму'
                onOk={form.submit}
                onCancel={closePopup}
                okText='Врахувати'
                cancelText='Скасувати'
              >
                <Form
                  form={form}
                  name='add_dish'
                  initialValues={{
                    remember: true,
                  }}
                  onFinish={onSubmit}
                >
                  <Row>
                    <Col flex={4}><div style={{fontSize: 18, marginTop:'6px
'}}>Скільки грам готового продукту Ви вжили?</div></Col>
                    <Col flex={1}>
                      <Form.Item name='quantity'>
                        <InputNumber size='large' min={100} max={5000} defa
ultValue={100} />
                      </Form.Item>
                    </Col>
                  </Row>
                </Form>
              </Modal>
            </div>
            <TagsSection tags={recipe.tags} />
            {comments && <CommentSection comments={comments} recipeId={reci
peId} />}
          </Content>
        </Layout>
      )}
```

```
    </>
  );
};
import React from 'react';
import { RecipeList, FilterMenu, AppBreadcrumb } from '../components';
import { RouteComponentProps } from '@reach/router';
import { Layout } from 'antd';
import { RECIPESLIST } from '../mocks';

const { Content, Sider } = Layout;

export const Recipes: React.FC<RouteComponentProps> = () => {
  let recipes = RECIPESLIST;

  return (
    <Layout>
      <Sider width={300} className='site-layout-background'>
        <FilterMenu />
      </Sider>
      <Layout style={{ padding: '0 24px 24px' }}>
        <AppBreadcrumb />
        <Content
          className='site-layout-background'
          style={{
            padding: 0,
            margin: 0,
            minHeight: 280,
          }}
        >
          <RecipeList recipes={recipes} />
        </Content>
      </Layout>
    </Layout>
  );
};

import React from 'react';
import { Typography, Rate, Space, Divider, Button, Carousel } from 'antd';
import { Icon } from '@iconify/react';
import { ClockCircleOutlined, HeartOutlined } from '@ant-design/icons';
import foodtrayIcon from '@iconify/icons-whh/foodtray';
import fireIcon from '@iconify/icons-el/fire';
import chefHat from '@iconify/icons-mdi/chef-hat';
import { eComplexity } from '../../entities/enum';

const { Paragraph } = Typography;

type tMainInfoSectionProps = {
  title: string;
  rating?: number;
  calories?: number;
  numberOfServings?: number;
  cookingTime?: number;
  complexity?: eComplexity;
  images?: string[];
  description?: string;
};

export const MainInfoSection: React.FC<tMainInfoSectionProps> = (props) =>
{
  return (
    <>
      <div style={{ textAlign: 'center' }}>
        <b style={{ fontSize: 48 }}>{props.title}</b>
```

```
        <p>
          {props.rating && <Rate disabled defaultValue={props.rating} style
={{ fontSize: 30 }} />}
        </p>
        <Space size='large'>
          <div>
            <Icon icon={fireIcon} style={{ fontSize: '14px' }} /> {props.ca
lories} ккал
          </div>
          {props.numberOfServings && (
            <div>
              <Icon icon={foodtrayIcon} style={{ fontSize: '12px' }} /> {pr
ops.numberOfServings}{' '}
              порцій
            </div>
          )}
          {props.cookingTime && (
            <div>
              <ClockCircleOutlined /> {props.cookingTime} хвилин
            </div>
          )}
          {props.complexity && (
            <div>
              <Icon icon={chefHat} style={{ fontSize: '14px' }} /> {props.c
omplexity}
            </div>
          )}
          <Divider type='vertical'/>
          <Button type='link' icon={<HeartOutlined />}>
            Зберігти
          </Button>
        </Space>
      </div>
      <Carousel
        style={{
          margin: 'auto',
          position: 'relative',
        }}
        autoplay
      >
        {props.images?.map((image) => {
          return (
            <div>
              <div
                style={{
                  width: '550px',
                  height: '550px',
                  margin: 'auto',
                  overflow: 'hidden',
                  position: 'relative',
                }}
              >
                <img
                  style={{
                    position: 'absolute',
                    margin: 'auto',
                    transform: ' translate(-50%, -50%)',
                    top: '50%',
                    left: '50%',
                    maxWidth: '100%',
                    maxHeight: '100%',
                  }}
                  src={image}
                  alt='Фотокартка рецепту'
```

```
                />
              </div>
            </div>
          );
        })}
      </Carousel>

      <Divider>Історія рецепту</Divider>
      <i>
        <Paragraph
          style={{
            width: '80%',
            margin: 'auto',
          }}
          ellipsis={{
            rows: 2,
            expandable: true,
          }}
        >
          {props.description}
        </Paragraph>
      </i>
      <Divider style={{ backgroundColor: 'blue' }} />
    </>
  );
};

import React from 'react';
import { InstructionStep } from '../../entities';
import { List, Button, Row, Col } from 'antd';
import { PrinterOutlined, ClockCircleOutlined } from '@ant-design/icons';

type tInstructionSectionProps = {
  instruction: InstructionStep[];
  cookingTime?: number;
};

export const InstructionSection: React.FC<tInstructionSectionProps> = (prop
s) => {
  const instruction = props.instruction.map((step, i) => ({
    title: `Крок ${i + 1}:`,
    content: step.description,
    image: step.image,
  }));

  return (
    <>
      <div style={{ marginTop: '5%' }}>
        <List
          header={
            <Row justify='space-
between'style={{ marginLeft: '0%', marginRight: '4%' }}>
              <Col>
                <h1 style={{ marginLeft: '15px', fontSize: '24px' }}>Інстру
кція приготування</h1>
              </Col>
              <Col>
                <div>
                  {props.cookingTime && (
                    <span style={{ fontSize: '16px' }}>
                      <ClockCircleOutlined /> {props.cookingTime} хвилин
                    </span>
                  )}
```

```
                  <Button size='large' type='link' icon={<PrinterOutlined />}>
                    Друкувати
                  </Button>
                </div>
              </Col>
            </Row>
          }
          itemLayout='vertical'
          size='large'
          dataSource={instruction}
          renderItem={(item) => (
            <List.Item key={item.title} extra={<img width={272} alt='logo' src={item.image} />}>
              <List.Item.Meta title={item.title} />
              <p style={{ fontSize: '243' }}>{item.content}</p>
            </List.Item>
          )}
        />
      </div>
    </>
  );
};
```

```
import React from 'react';
import { Tag, Divider } from 'antd';

const colors = [
    'magenta',
    'red',
    'volcano',
    'orange',
    'gold',
    'lime',
    'green',
    'cyan',
    'blue',
    'geekblue',
    'purple',
  ];

  function getColor() {
    return colors[Math.floor(Math.random() * colors.length)];
  }

type tTagsSectionProps = {
  tags?: string[];
};

export const TagsSection: React.FC<tTagsSectionProps> = (props) => {
  return (
    <>
     <Divider orientation="left" plain>
      Теги
    </Divider>
      <div style={{ margin: '1%' }}>
          {props.tags && props.tags.map((tag) => <Tag color={getColor()}>{tag}</Tag>)}
        </div>
    </>
  );
```

```
};

 import React from 'react';
import { Table } from 'antd';

const { Column } = Table;

type tEnergyValueSectionProps = {
  caloricity: number;
  protein: number;
  triglyceride: number;
  carbohydrate: number;
};

export const EnergyValueSection: React.FC<tEnergyValueSectionProps> = (prop
s) => {
  return (
    <>
      <div style={{ textAlign: 'center', marginTop: '5%' }}>
        <h1>Енергетична цінність на порцію</h1>
        <Table
          dataSource={[
            {
              caloricity: `${props.caloricity} ккал`,
              protein: `${props.protein} грам`,
              triglyceride: `${props.triglyceride} грам`,
              carbohydrate: `${props.carbohydrate} грам`,
            },
          ]}
          pagination={false}
        >
          <Column align='center' title='Калорійність' dataIndex='caloricity
' key='caloricity' />
          <Column align='center' title='Білки' dataIndex='protein' key='pro
tein' />
          <Column align='center' title='Жири' dataIndex='triglyceride' key=
'triglyceride' />
          <Column align='center' title='Вуглеводи' dataIndex='carbohydrate'
 key='carbohydrate' />
        </Table>
      </div>
      <div style={{ textAlign: 'right', marginRight:'20px' }}>
        <i>* Калорійність розрахована для сирих продуктів</i>
      </div>
    </>
  );
};
```

Додаток З
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS

Виконав: Карпенко О. О.

Керівник: Доц. кафедри ПЗКС, к.т.н., доц., Заболотня Т. М.

Київ – 2020

АКТУАЛЬНІСТЬ

- Сучасний ритм життя не залишає часу на самостійне планування свого раціону.
- Переважна більшість людей харчується незбалансовано.
- Наявні рішення обмежені в виборі рецептів та не є цілком автоматизованими.
- Більшість рецептів не містить інформацію про БЖВ готової страви.
- Розробка веб-додатку для підтримки формування раціону дозволить користувачам збалансувати свій раціон.

ІСНУЮЧІ АНАЛОГИ

Cara: Food, Mood, Poop Tracker

Веб-сайт «Еда»

Rise Up + Recover

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: спростити процес формування раціону користувачів за допомогою розроблення автоматизованої системи генерації рекомендацій рецептів страв.

Завдання:

1. Проаналізувати аналогічні програмні рішення присутні на ринку.
2. Розробити веб-додаток для автоматизованого формування рекомендацій щодо збалансованого щоденного раціону користувача на основі його вподобань та потреб
3. Розгорнути веб-додаток в оточені платформи хмарних обчислень Amazon Web Services (AWS).
4. Провести тестування ПЗ.

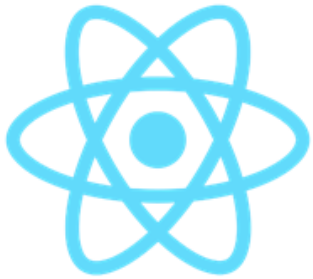
ОБРАНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Мова програмування C#

СКБД PostgreSQL

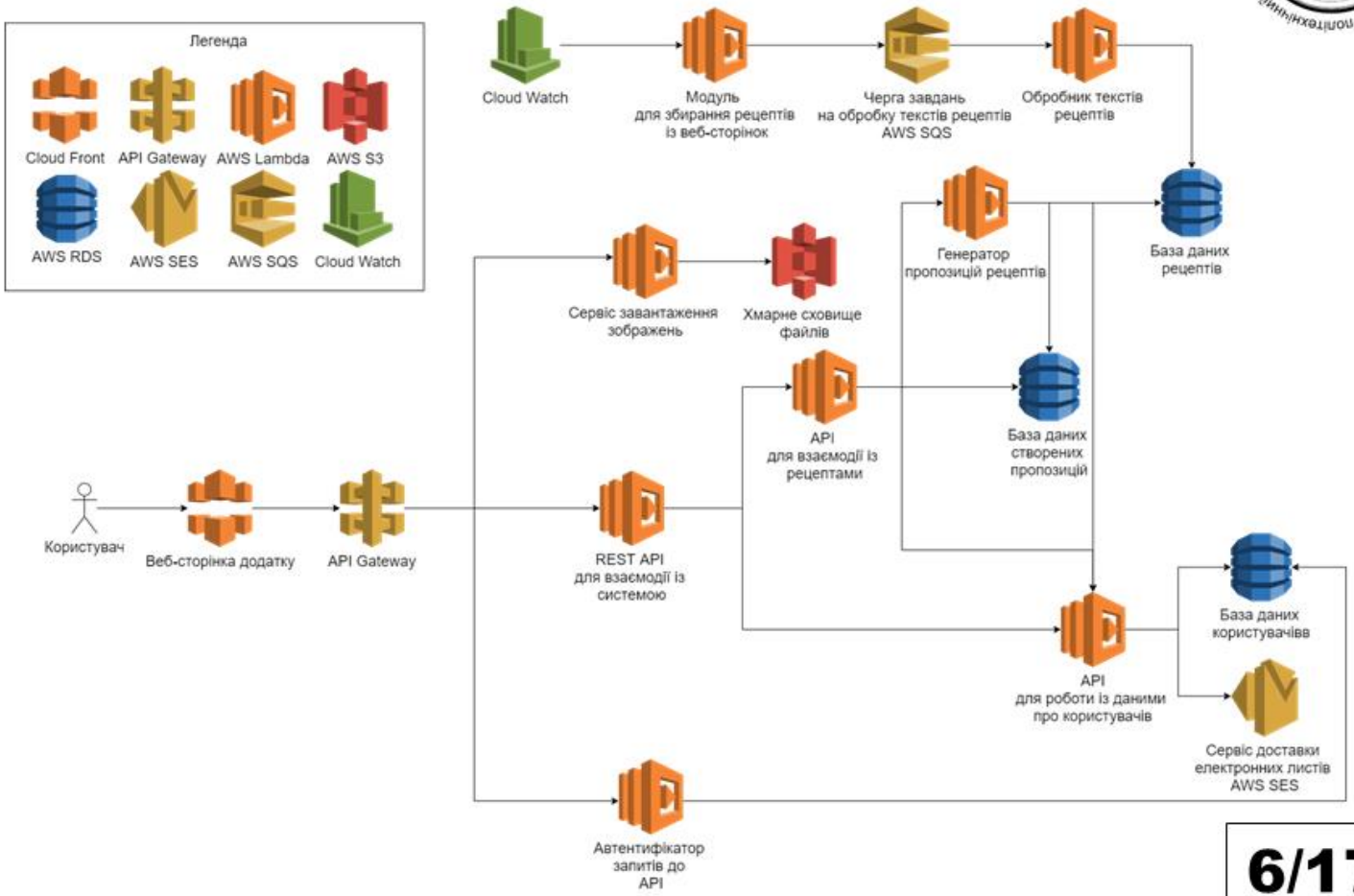
Платформа хмарних обчислень Amazon Web Services (AWS)

Мова програмування TypeScript

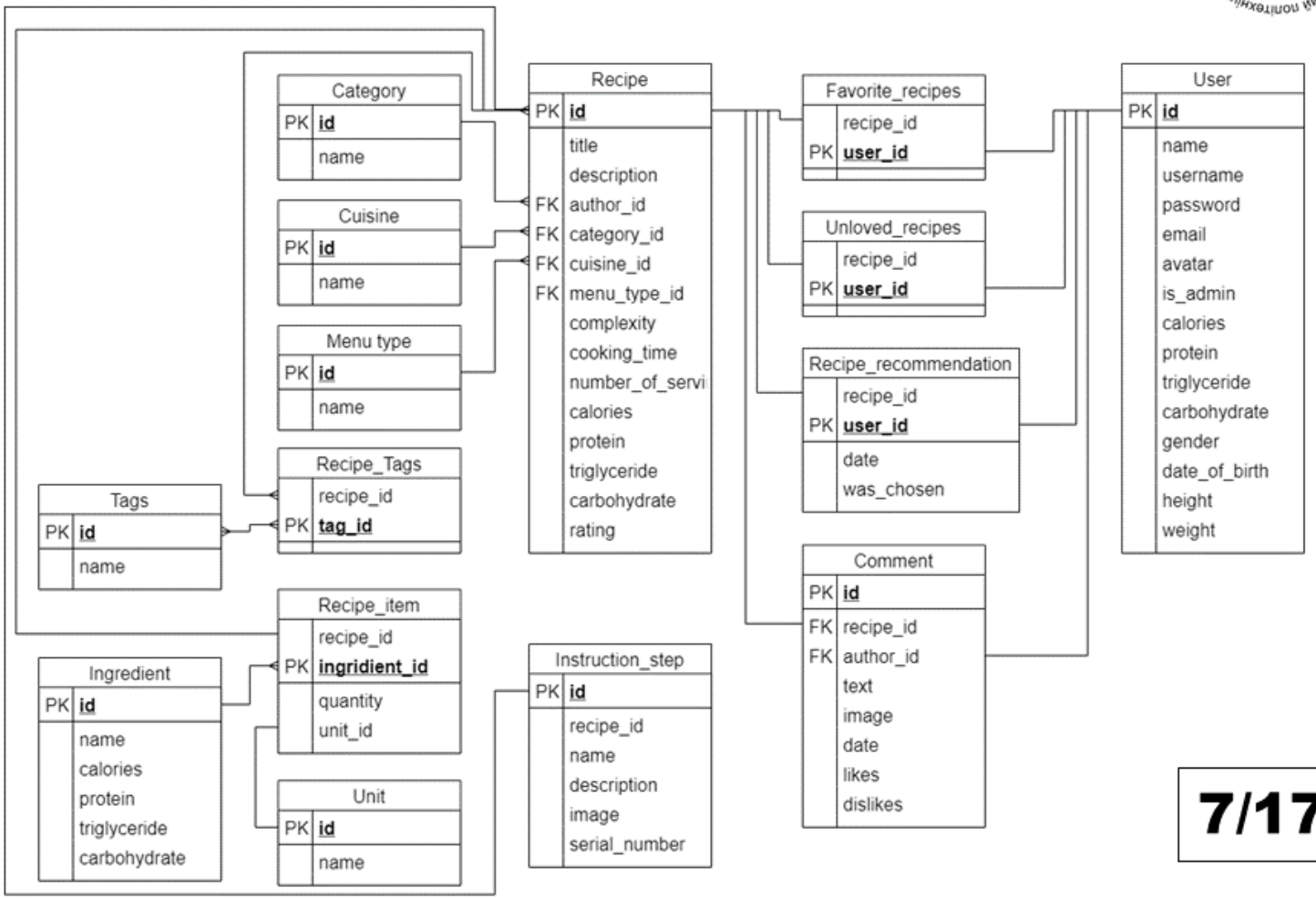


Клієнтська бібліотека React

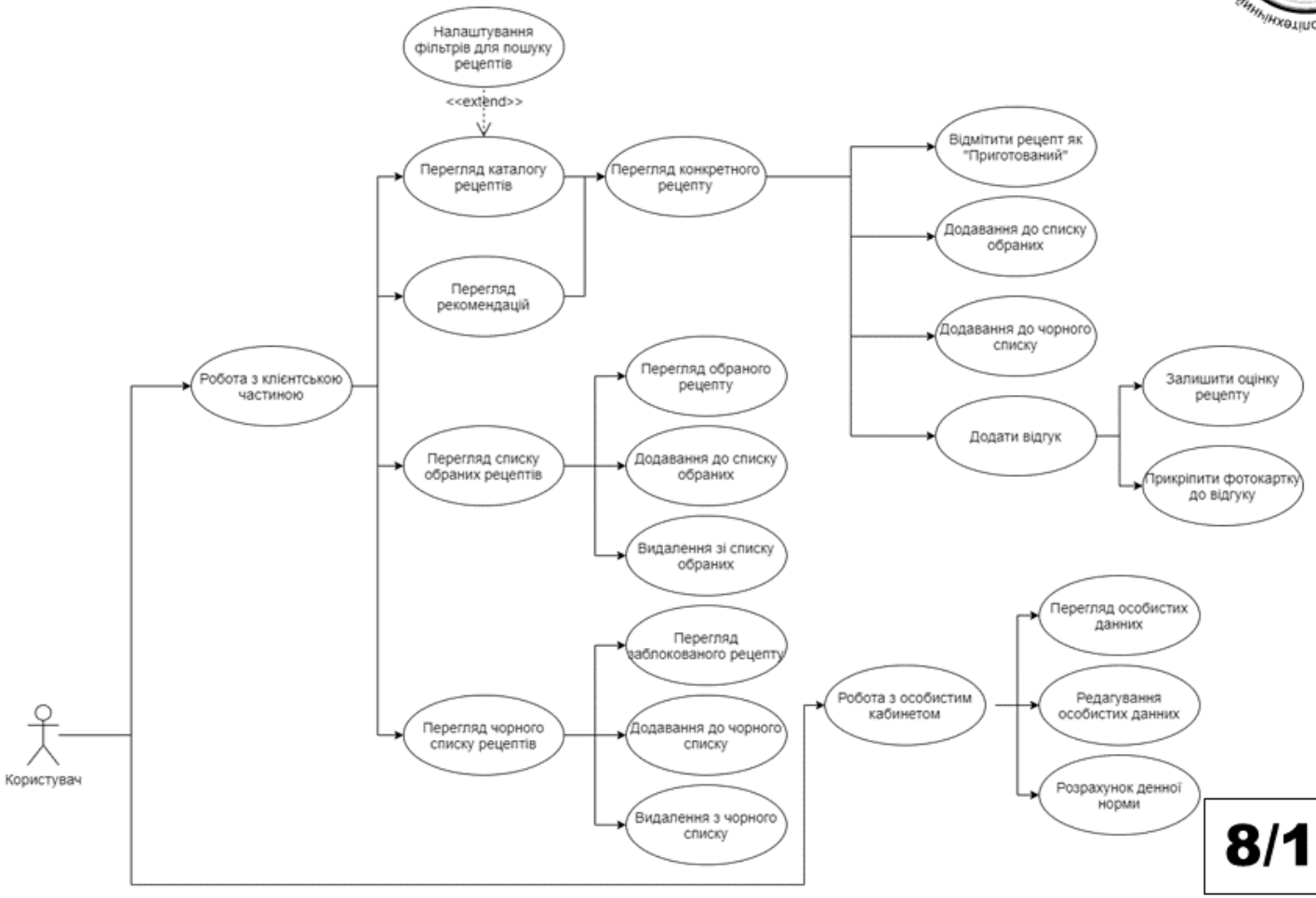
АРХІТЕКТУРА СИСТЕМИ



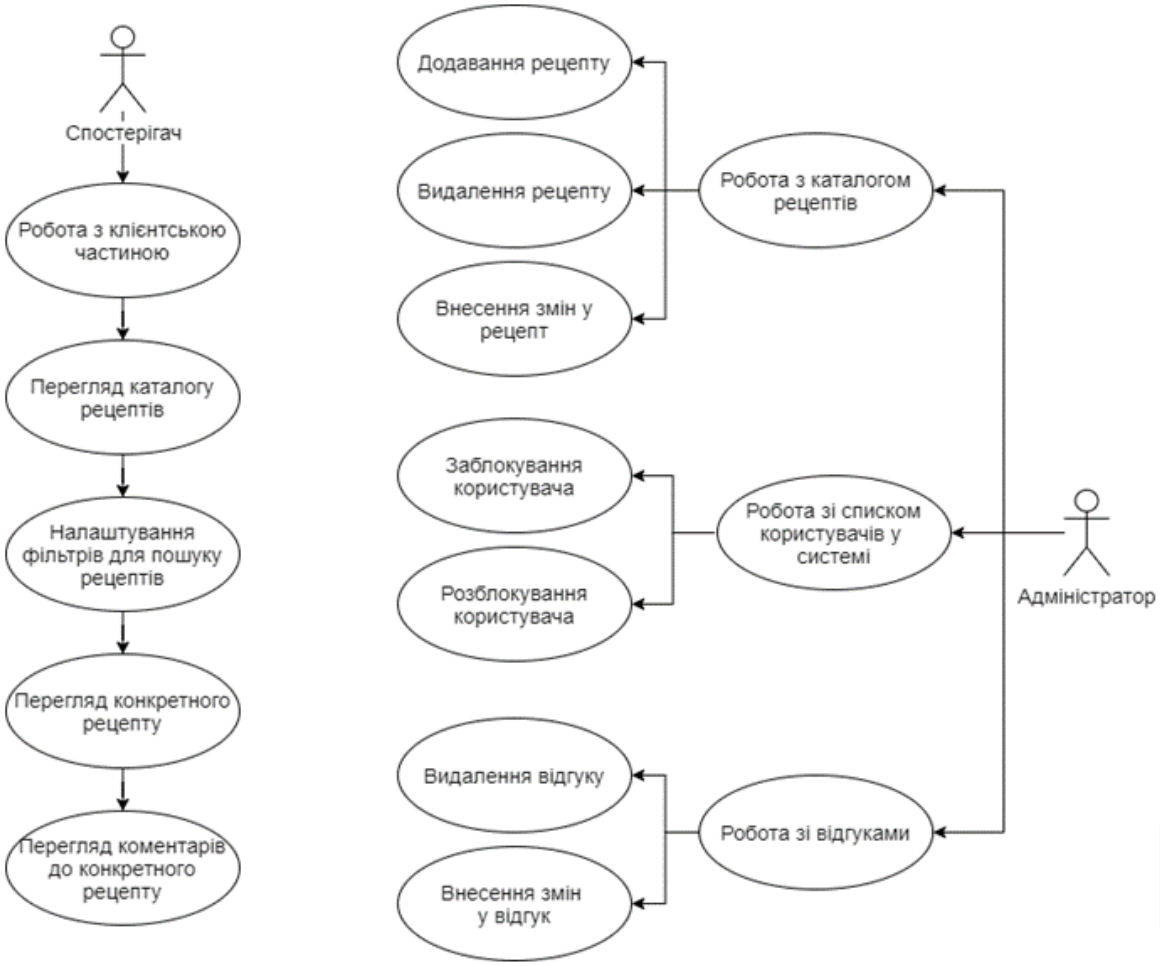
СТРУКТУРА БД



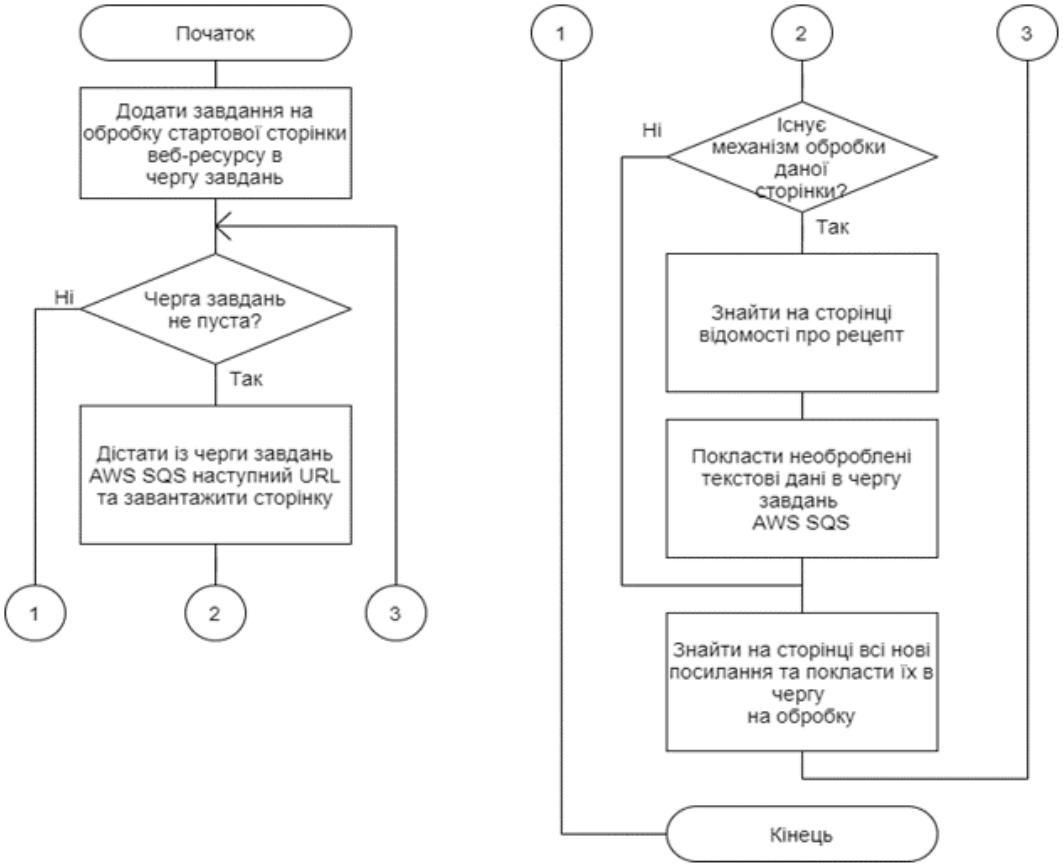
ДІАГРАМА ПРЕЦЕДЕНТІВ 1



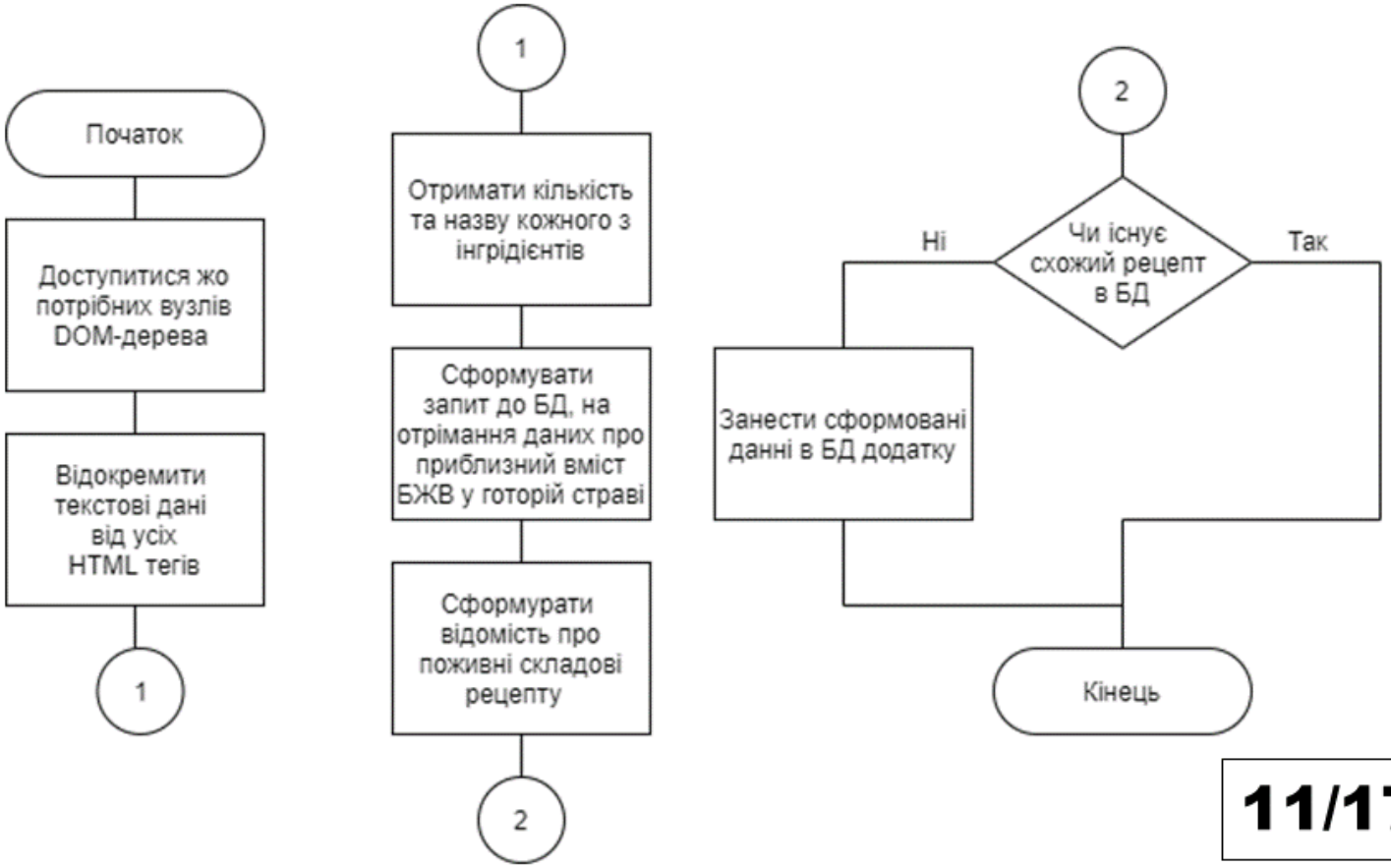
ДІАГРАМА ПРЕЦЕДЕНТІВ 2



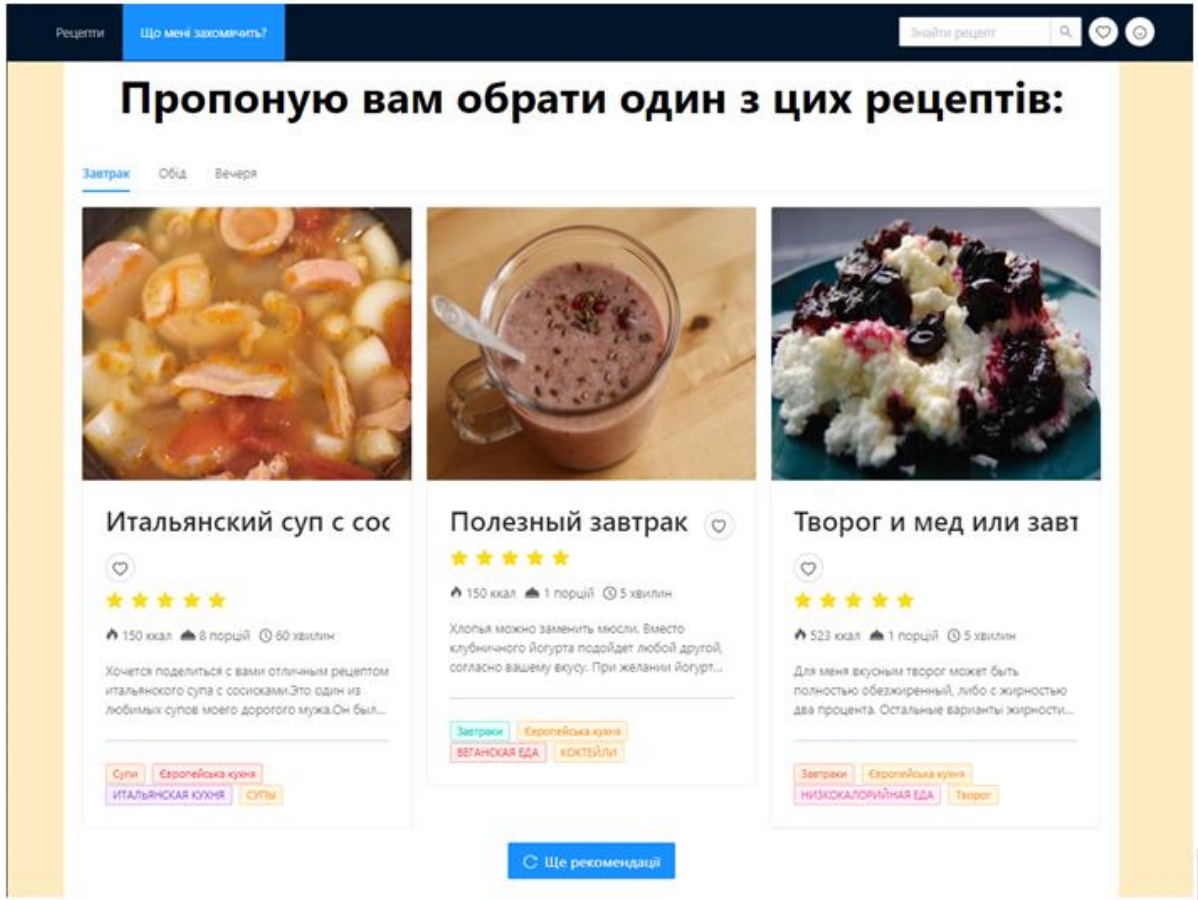
АЛГОРИТМ ОТРИМАННЯ ДАНИХ РЕЦЕПТІВ ІЗ ВЕБ-СТОРІНОК



АЛГОРИТМ ОБРОБЛЕННЯ ДАНИХ РЕЦЕПТІВ



ІНТЕРФЕЙС КОРИСТУВАЧА



ІНТЕРФЕЙС КОРИСТУВАЧА

Завтрак для ленивых

87 ккал | 3 порцій | 15 хвилин | Рецепти для чайників | Зберігти

Енергетична цінність на порцію

Калорійність	Білки	Жири	Вуглеводи
87 ккал	8.1 грам	2.3 грам	8.3 грам

** Калорійність розрахована для сирих продуктів*

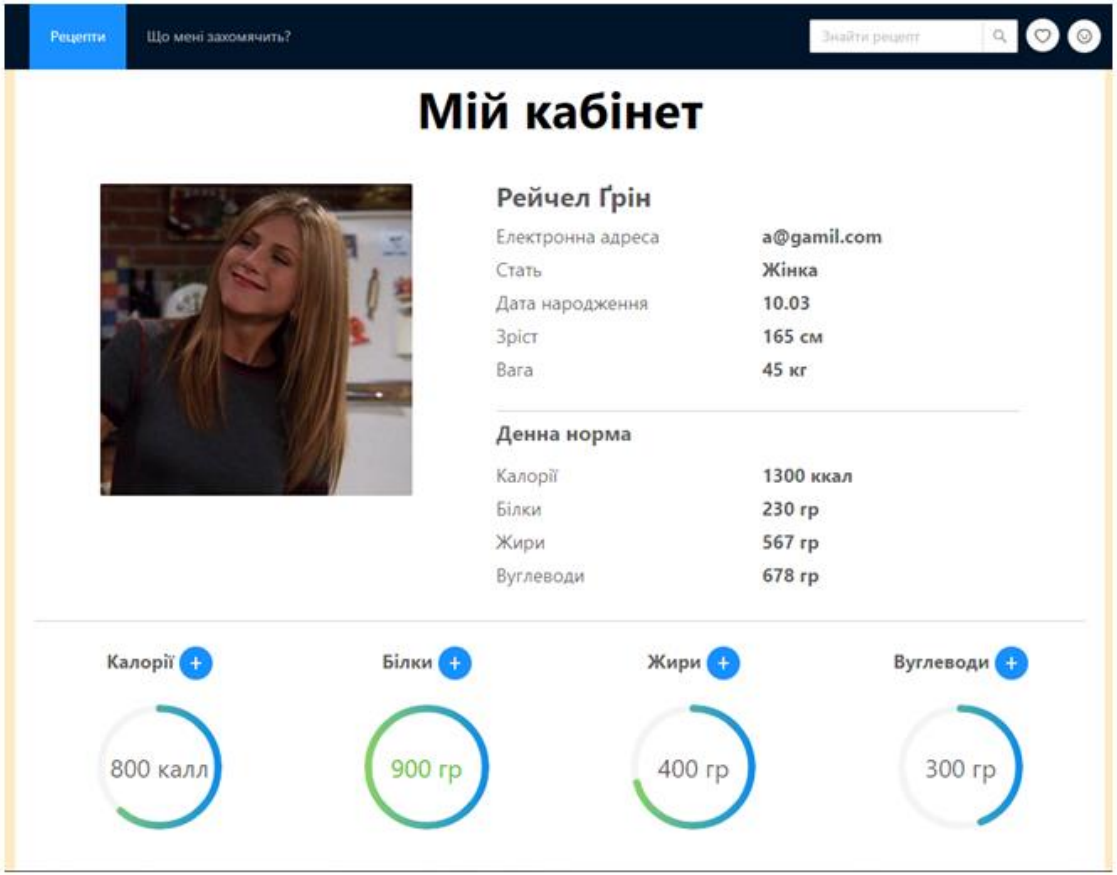
Інгредієнти на 6 порцій

Інгредієнти	Кількість
Яйцо куриное	1 шт.
Мягкий творог	200 гр.
Пшеничная мука	30 гр.
Сахар	1 чайн. л.
Соль	1 чайн. л.

Історія рецепту

Ленивые вареники — те же галушки, только сладкие и выступающие сольно. Готовятся не дольше любой каши и в отличие от любой каши не вгоняют по утрам своим видом в тоску. Их можно подавать с любимым вареньем. ... Expand

ІНТЕРФЕЙС КОРИСТУВАЧА



АПРОБАЦІЯ

1. V Міжнародна науково-практична конференція “Інформаційні технології в освіті, науці й техніці ” (ІТОНТ-2020). Результати опубліковані у вигляді тез доповіді.

2. Підготовлено пакет документів на отримання авторського свідоцтва.

ВИСНОВКИ

1. Проведено аналіз доступних програмних рішень.
2. Розроблено архітектуру веб-додатку та БД.
3. Реалізовано серверну та клієнтську частини веб-додатку.
4. Розгорнуто веб-додаток в оточені платформи хмарних обчислень Amazon Web Services (AWS).
5. Розроблено алгоритми отримання даних рецептів з веб-сторінки та обробки даних рецептів.
6. Реалізовано вимоги до ПЗ.
7. Протестовано роботу веб-додатку та порівняно з аналогами.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олена КАРПЕНКО

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-додаток для підтримки формування раціону користувача на основі платформи AWS, який являє собою веб-додаток, створений на платформі Microsoft .Net Core та AWS з використанням технології React JS.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок web-ресурсу;
- наявність доступу до бази данних рецептів;
- забезпечення належного рівня безпеки даних;
- зручність роботи з веб-додатком;
- відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

- функціональне тестування, зокрема на рівні Critical path test (базове тестування);
- тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
- тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність веб-додатку перевіряється шляхом:

- динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
- динамічного ручного тестування на відповідність функціональним вимогам;
- статичного тестування коду;
- тестування веб-додатку в різних веб-браузерах;
- тестування при максимальному навантаженні;
- тестування стабільності роботи при різних умовах;
- тестування зручності використання;
- тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ ПІДТРИМКИ ФОРМУВАННЯ РАЦІОНУ КОРИСТУВАЧА НА ОСНОВІ ПЛАТФОРМИ AWS

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олена КАРПЕНКО

2020

ЗМІСТ

1. Опис структури веб-додатку

Веб-додаток для сприяння формування раціону користувача на основі платформи AWS складається із статичних і динамічних веб-сторінок, вміст яких формується в прямо у вікні браузеру та залежить від даних, отриманих в результаті запиту до веб-серверу за допомогою технології AJAX. Веб-ресурс є одномовним із українською мовою інтерфейсу.

До статичних сторінок веб-додатку належать:

- головна сторінка веб-додатку.

До сторінок веб-додатку належать сторінки:

- головна сторінка перегляду всіх рецептів;
- сторінка рекомендацій рецептів користувачу;
- сторінка редагування профілю користувача;
- сторінка відновлення паролю;
- сторінка перегляду рецепту;
- сторінка авторизації користувачів;
- сторінка реєстрації користувачів;
- сторінка перегляду обраних рецептів користувачем.

Кожна веб-сторінка зверху містить шапку. В ній відображаються наступні елементи: посилання на сторінку пошуку рецептів, інформація про авторизованого користувача або кнопки логіну та реєстрації, натискаючи на які, користувач викликає модальні вікна із формами для авторизації або реєстрації а також поле вводу для пошуку рецепту за назвою.

2. Опис вмісту статичних веб-сторінок

Головна сторінка веб-додатку виступає рекламним банером, та надає відвідувачам інформацію про призначення та принципи роботи цього веб-сайту. На ній розміщено кнопку реєстрації та поле пошуку рецептів за назвою.

Після натискання на кнопку «Шукати» користувач буде направлений на сторінку перегляду результатів пошуку рецептів. Після натискання на

кнопку «Увійти» користувач побачить спливаюче вікно із формою авторизації.

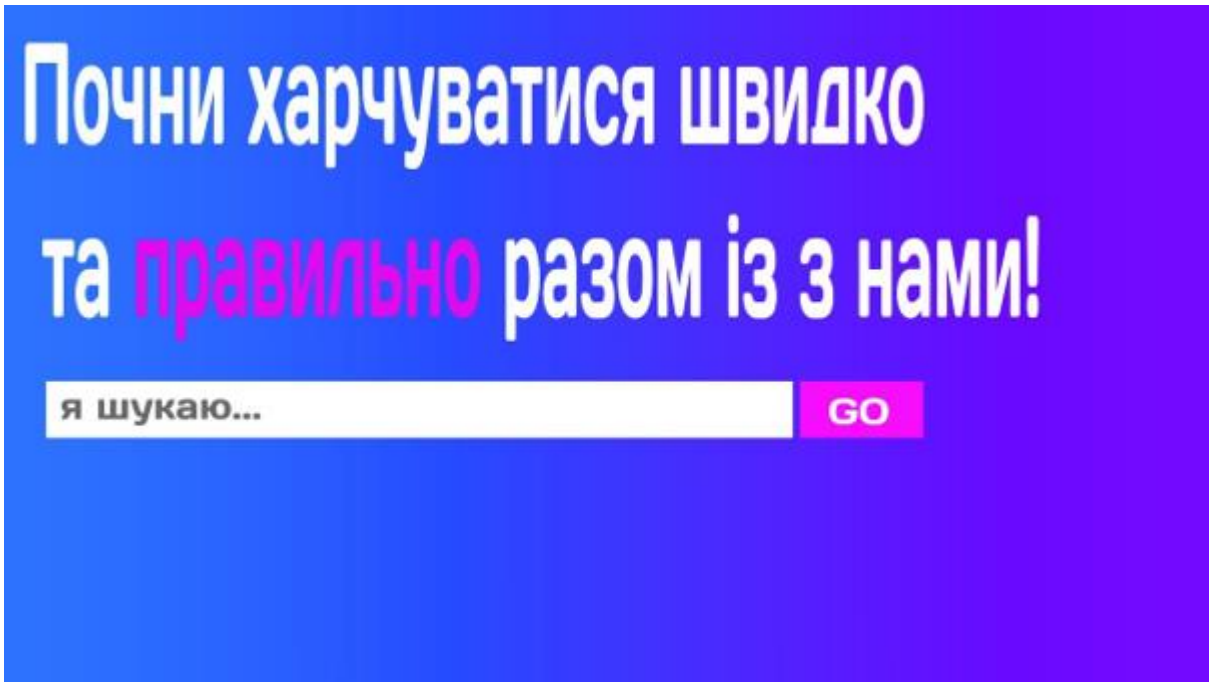


Рис. 1. Головна сторінка веб-додатку

3. Процедура авторизації користувача

Для авторизації та реєстрації користувача було реалізовано спливаючі вікна, за допомогою яких він може увійти в свій профіль без необхідності відкривати додаткові сторінки. Для забезпечення безпеки даних відвідувача, який вводить свій пароль, було додано чек-бокс, який робить його поле вводу паролю відкритим. Інакше в цьому полі відображаються зірочки замість реальних літер. У вікні авторизації є посилання на форму для відновлення паролю.

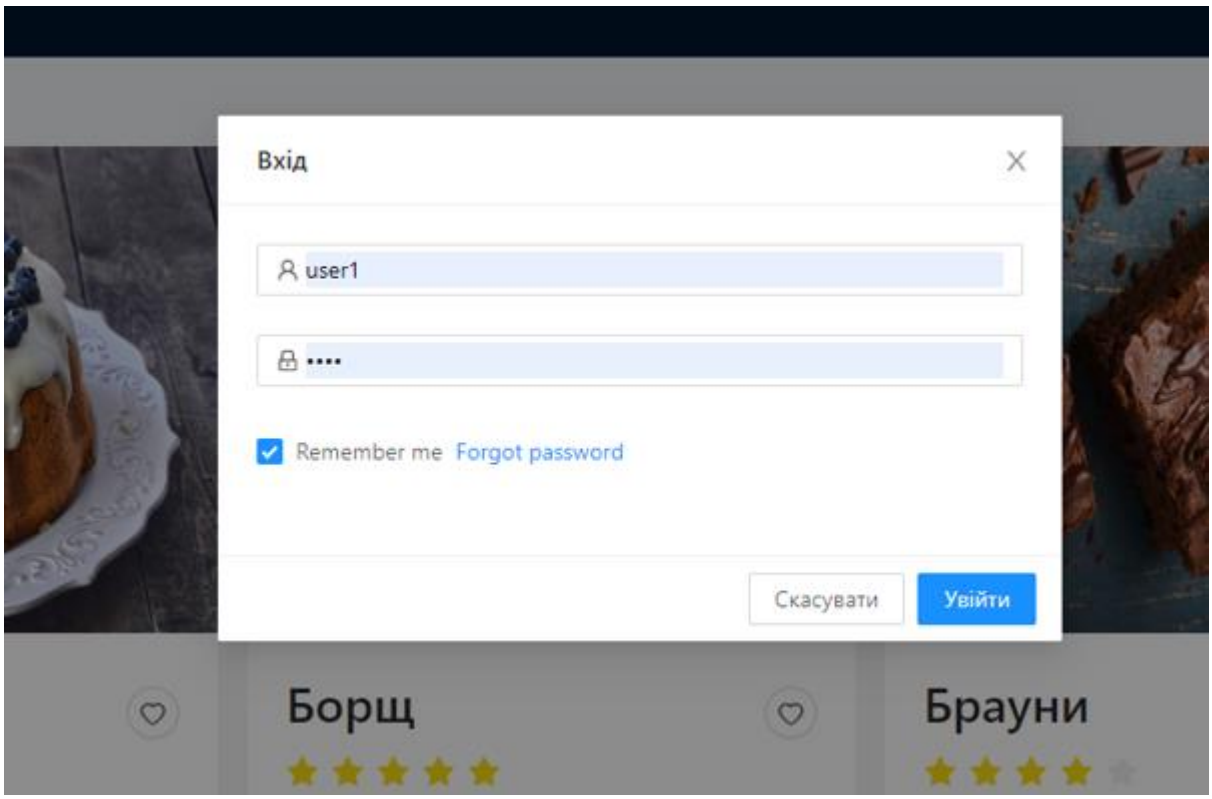


Рис. 2. Вспливаюче вікно авторизації

Якщо користувач забув свій пароль, він може скористатися формою відновлення паролю. Для цього йому достатньо перейти за посиланням, яке є у вікні авторизації та ввести свою електрону пошту. На неї прийде лист (рис. 3) із посиланням на сторінку, де користувач може змінити пароль.

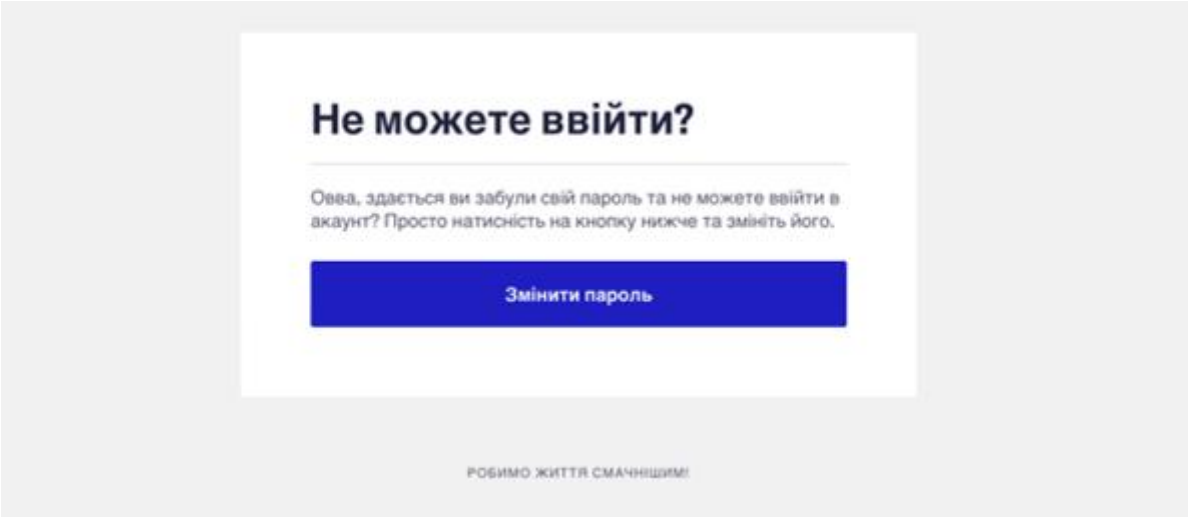


Рис. 3. Лист із посиланням на форму відновлення паролю

4. Сторінка пошуку рецептів користувачем

Основна сторінка, з якою взаємодіють користувачі веб-додатку – сторінка пошуку рецептів (рис. 4). На ній відображаються результати фільтрації наявних в БД додатку рецептів за критеріями, які обрав користувач.

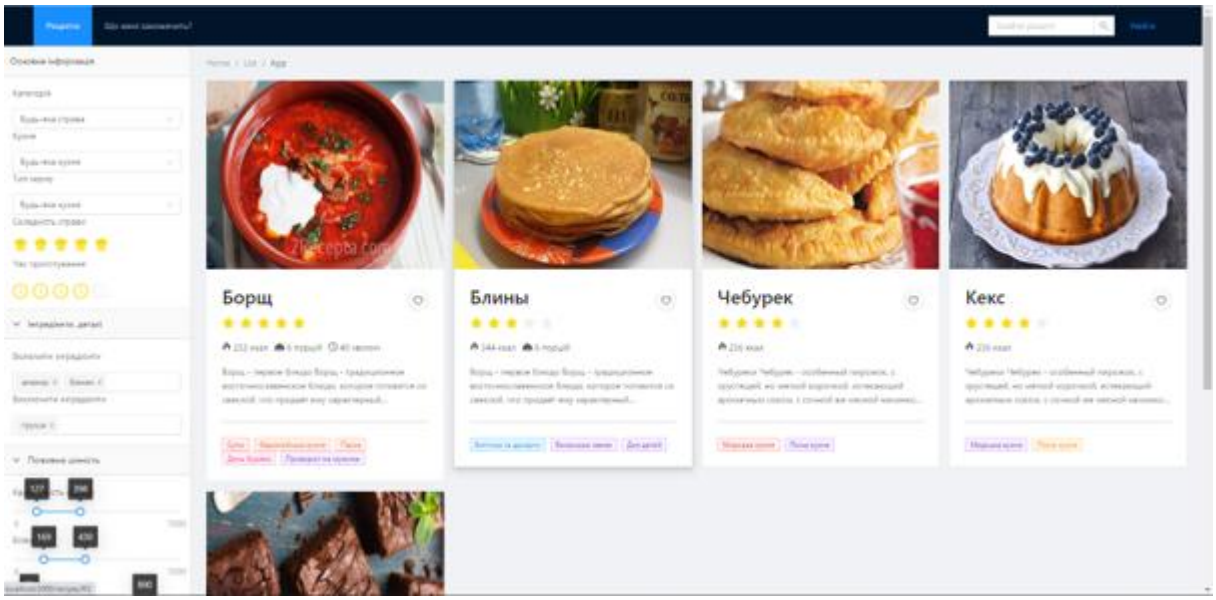


Рис. 4. Сторінка пошуку рецептів

Для того аби задавати критерії пошуку рецептів користувачу пропонується обрати їх у меню зліва. Там розміщено основні елементи керування фільтрацією рецептів. До них відносяться: категорія, країна походження, тип меню, складність, час приготування, калорійність, обов’язкові та заборонені інгредієнти, БЖВ.

Після застосування змін у критеріях фільтрації користувач побачить зміни на сторінці, без її перезавантаження.

По центру сторінки розміщено зону виводу результатів пошуку. Кожен рецепт представляє собою карточку із короткою інформацією про рецепт. Зокрема в цих блоках розміщено такі дані:

- калорійність страви;
- очікувана кількість порцій;

- приблизний час приготування страви;
- назва страви;
- фотографія із готовою стравою;
- зірочки, які відображають рейтинг рецепту на основі відгуків;
- теги, прив’язані до рецепту;
- кнопка додавання в «обране».

Також в системі є сторінка перегляду обраних рецептів. Вона ідентична до сторінки пошуку рецептів, окрім того, що на ній до критеріїв пошуку додається додатковий параметр – вони мають бути в обраному поточного користувача.

5. Сторінка рекомендацій рецептів користувачу

На сторінці рекомендацій рецептів користувачу відображається перелік із трьох страв, які на думку системи відповідають смакам користувача. Карточки із кожною стравою є такими самими як і на сторінці пошуку рецептів. Якщо користувачеві не подобається підборка рецептів, він може натиснути на кнопку «ще рекомендації». В такому разі йому буде представлена наступна трійка рецептів.

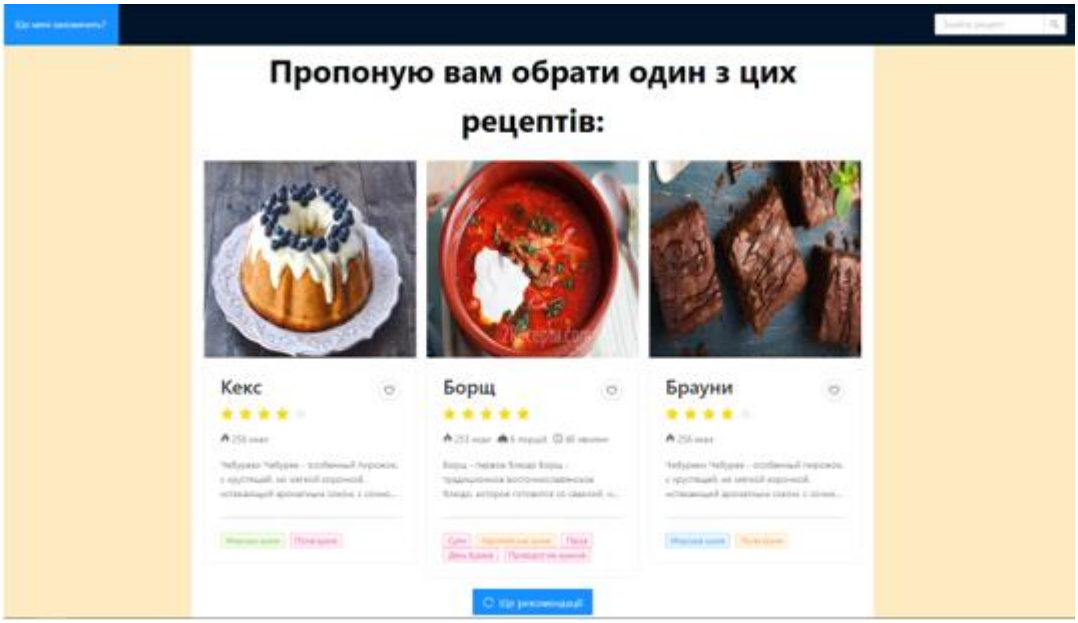


Рис. 5. Сторінка рекомендацій страв користувачу

5. Сторінка перегляду рецепту

Після того, як користувач знайшов рецепт, який його цікавить, він може перейти на сторінку перегляду розширеної інформації про нього.

На цій сторінці (рис. 6) відображається карусель із всіма наявними фотографіями рецепту, його детальний опис та покрокова інструкція приготування, розрахунок калорійності та вмісту БЖВ, коментарі від авторизованих користувачів.

Зверху над фотографією відображається та сама інформація, яка була доступна на картці із короткими відомостями про рецепт:

- калорійність;
- очікувана кількість порцій;
- час приготування;
- кнопка додавання до обраного.

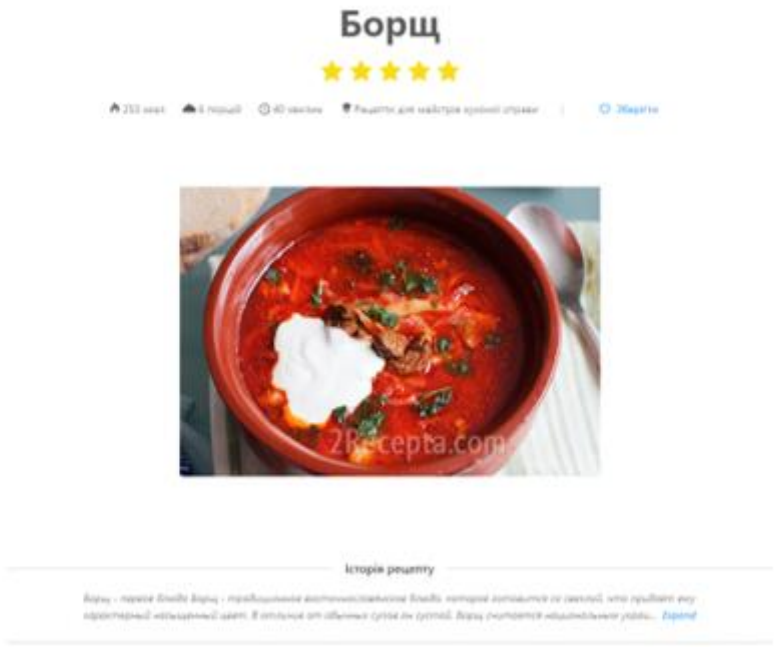


Рис. 6. Сторінка перегляду рецепту. Фотографія та опис страви

Нижче, після блоку із фотографіями готової страви розміщено таблицю із інгредієнтами, які потрібні для приготування рецепту (рис. 7).

Окрім цього поряд наведено розрахунок енергетичної цінності на порцію готової страви.

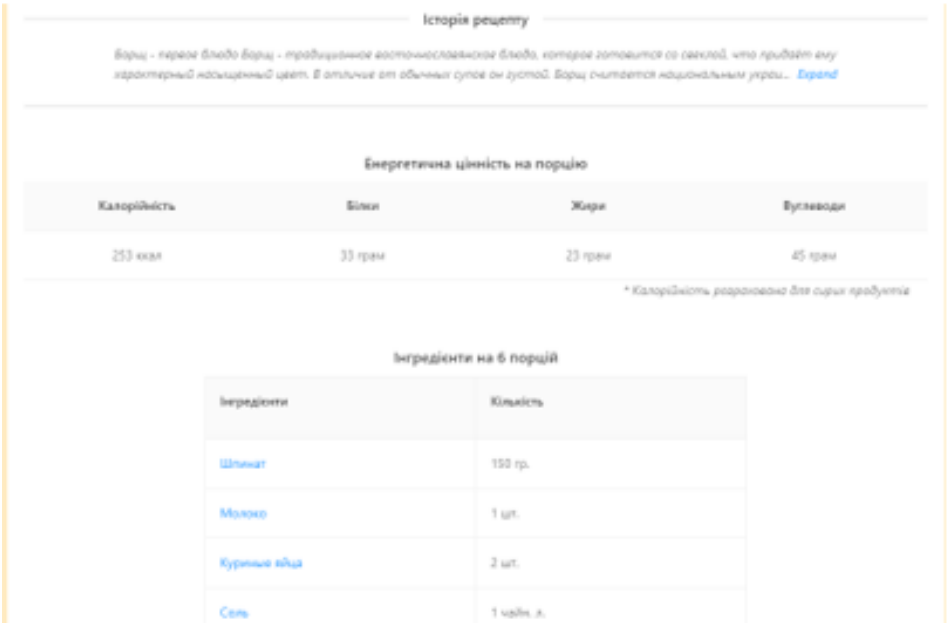


Рис. 7. Сторінка перегляду рецепту. Блок інгредієнтів

Інструкцію приготування візуальні розділено на блоки для зручності (рис. 8). До кожного блоку додане зображення, яке описує проміжний результат, який має бути отриманий на цьому кроці.

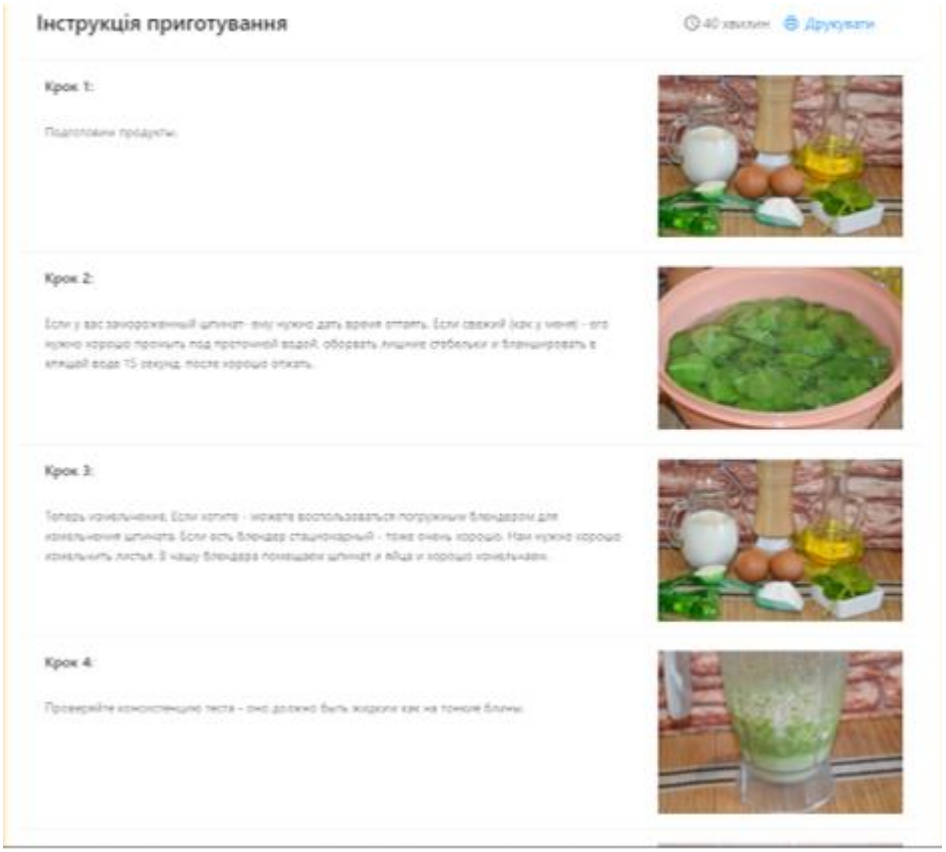


Рис. 8. Сторінка перегляду рецепту. Покрокова інструкція приготування

Під описом рецепту розміщено блок із коментарями користувачів (рис. 9). Кожен коментар складається з таких елементів:

- фотографія автора;
- ім’я автора;
- дата створення коментаря;
- текст коментаря;
- прикріплена до коментаря фотографія.

Поряд, якщо користувач авторизований, відображається форма для додавання нових коментарів. Вона складається із поля вводу тексту коментаря, кнопки для додавання фотографії та кнопки «відправити».

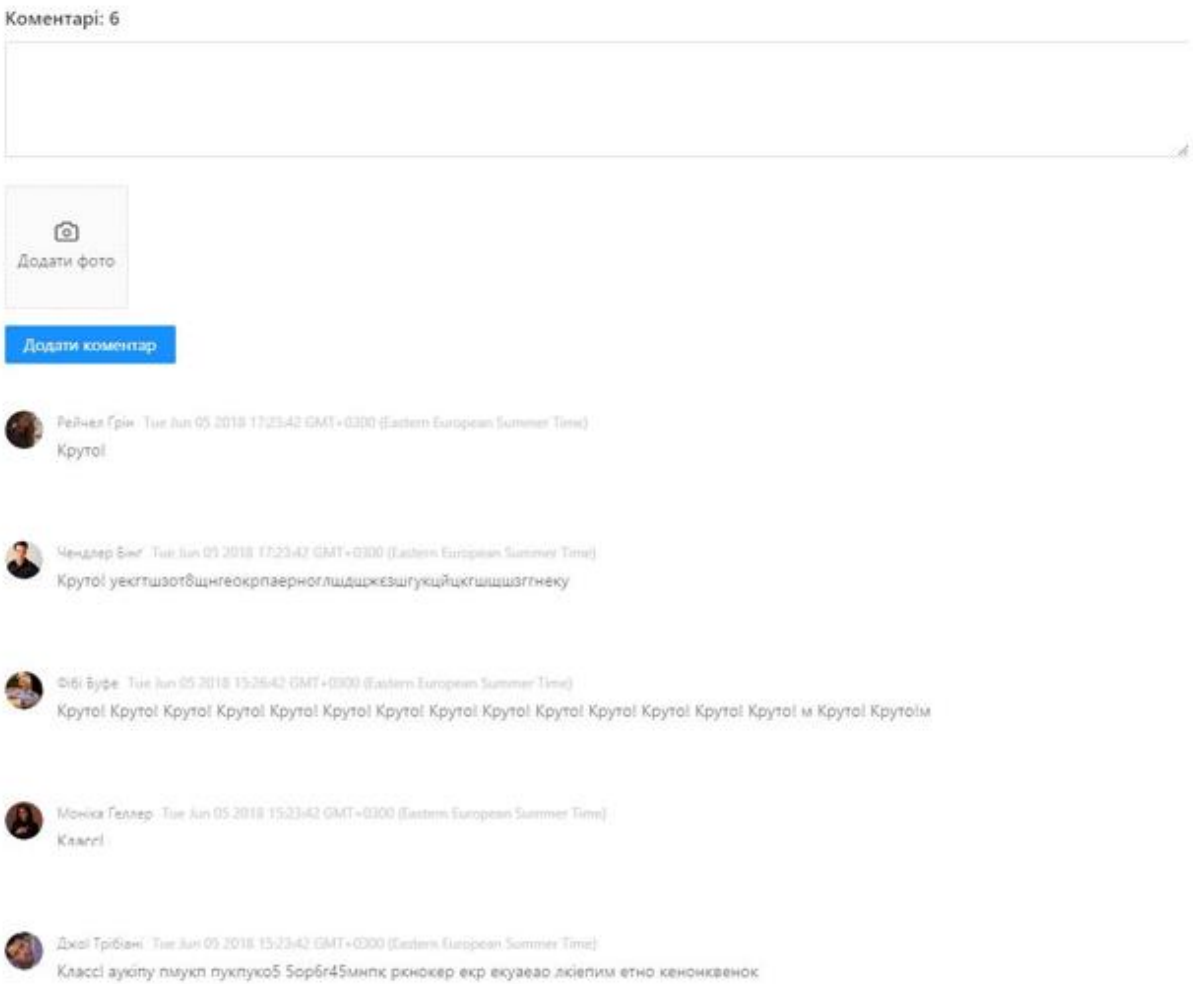


Рис. 9. Сторінка перегляду рецепту. Блок коментарів